```
PL/M-80 COMPILER    CPM 3.0 --- SET 1.3


ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SET
OBJECT MODULE PLACED IN SET.OBJ
COMPILER INVOKED BY:  PLM80 SET.PLM DEBUG OPTIMIZE PAGEWIDTH(132)



            $ TITLE('CPM 3.0 --- SET 1.3')

            /* MULTI FILE INPUT VERSION   11/11/82 */
            /* took out call passwd in readlabel */
            /* added test for NONBANK in password, protect and default  11/19/82 */
            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                    * * *  SET  * * *


             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


  1         set:
            do;

  2  1      declare
            mpmproduct literally '01h', /* requires mp/m */
            cpmversion literally '30h'; /* requires 3.0 cp/m */


  3  1      declare
            true        literally '1',
            false       literally '0',
            dcl         literally 'declare',
            lit         literally 'literally',
            proc        literally 'procedure',
            addr        literally 'address',
            tab         literally '9',
            cr          literally '13',
            lf          literally '10',
            ctrlc       literally '3h',
            ctrlx       literally '18h',
            ctrlh       literally '8h';

  4  1      declare
                  opt$access      literally '0',
                  opt$archive     literally '1',
                  opt$create      literally '2',
                  opt$default     literally '3',
                  opt$dir         literally '4',
                  opt$f1          literally '5',
                  opt$f2          literally '6',
                  opt$f3          literally '7',
                  opt$f4          literally '8',
                  opt$name        literally '9',
                  opt$pass        literally '10',
                  opt$prot        literally '11',
                  opt$ro          literally '12',
                  opt$rw          literally '13',
```

CP/M - PLUS V 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3 - 135

```
                opt$sys         literally '14',
                opt$update      literally '15',
                opt$page        literally '16',
                opt$nopage      literally '17',

                PERIOD          literally '02eh',
                PAGE            byte initial(false);

5  1    declare pla label public;

6  1    declare copyright (*) byte data (
          ' Copyright (c) 1982 Digital Research ');

        /*
                Digital Research
                Box 579
                Pacific Grove, Ca
                93950
        */
```

COPYRIGHT © 1932
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
$ eject
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  MESSAGES  * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


7  1            declare
                   not$found  (*) byte data (' File not found',0),
                   no$space   (*) byte data (' or no directory space',0),
                   invalid    (*) byte data ('Invalid file name.',0),
                   dirlabel   (*) byte data ('Directory Label ',0),
                   option$set (*) byte data (' attribute set ',0),
                   read$only  (*) byte data ('Read Only',0),
                   ro         (*) byte data (' (RO)',0),
                   read$write (*) byte data ('Read Write (RW)',0),
                   comma      (*) byte data (', ',0),
                   set$to     (*) byte data ('set to ',0),
                   error$msg  (*) byte data ('ERROR: ',0),
                   readmode   (*) byte data ('READ',0),
                   writemode  (*) byte data ('WRITE',0),
                   deletemode (*) byte data ('DELETE',0),
                   nopasswd   (*) byte data ('NONE',0),
                   on         (*) byte data ('    on   ',0),
                   off        (*) byte data ('    off  ',0),
                   label$name (*) byte data ('LABEL');

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  CP/M INTERFACE * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


8  1      declare
             maxb      address external,    /* addr field of jmp BDOS */
             fcb (33)  byte external,       /* default file control block */
             buff(128) byte external,       /* default buffer */
             buffa     literally '.buff',   /* default buffer */
             fcba      literally '.fcb',    /* default file control block */
             user$code byte;                /* current user code */


          /*            Routines used in SET for CPM 3.0              */

             /* reset drive mask */
9  1         declare reset$mask (16) address data (
               0000000000000001b,
```

```
                0000000000000010b,
                0000000000000100b,
                0000000000001000b,
                0000000000010000b,
                0000000000100000b,
                0000000001000000b,
                0000000010000000b,
                0000000100000000b,
                0000001000000000b,
                0000010000000000b,
                0000100000000000b,
                0001000000000000b,
                0010000000000000b,
                0100000000000000b,
                1000000000000000b );

10   1      boot: procedure external;
                /* reboot */
11   2          end boot;

12   1      mon1: procedure(f,a) external;
13   2          declare f byte, a address;
14   2          end mon1;

15   1      mon2: procedure(f,a) byte external;
16   2          declare f byte, a address;
17   2          end mon2;

18   1      declare mon3 literally 'mon2a';

19   1      mon3: procedure(f,a) address external;
20   2          declare f byte, a address;
21   2          end mon3;

                /********** SYSTEM FUNCTION CALLS **********************/

22   1      printchar: procedure(char);
23   2          declare char byte;
24   2          call mon1(2,char);
25   2          end printchar;

26   1      printb: procedure;                  /* print blank character */

27   2          call printchar(' ');

28   2      end printb;

29   1      printx: procedure(a);
30   2          declare a address;
31   2          declare s based a byte;
32   2              do while s <> 0;
33   3              call printchar(s);
34   3              a = a + 1;
35   3              end;
36   2          end printx;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 37  1     check$con$stat: procedure byte;

 38  2        return mon2(11,0);                /* console ready */

 39  2     end check$con$stat;


 40  1     crlf2: procedure;

 41  2           call printchar(cr);
 42  2           call printchar(lf);

 43  2     end crlf2;


 44  1     terminate: procedure;
 45  2        call crlf2;
 46  2        call mon1 (0,0);
 47  2     end terminate;


 48  1     crlf: procedure;
 49  2          declare charin     byte;

 50  2          if PAGE then do;
 52  3             line$out = line$out + 1;              /* output > page size ? */
 53  3             if line$out + 2 > line$page then do;
 55  4                    call crlf2;
 56  4                    call crlf2;
 57  4                    call printx(.('Press RETURN to continue.',0));

 58  4                    do while not check$con$stat;
 59  5                    end;

 60  4                    charin = mon2(1,0);              /* read character */
 61  4                    if charin = ctrlc then call terminate;
 63  4                    line$out = 1;
 64  4                    call crlf2;
 65  4               end;
 66  3          end;

 67  2             call crlf2;

 68  2     end crlf;

 69  1     print: procedure(a);    /* print the string starting at address a until the
                                     next 0 is encountered */
 70  2        declare a address;

 71  2        call crlf;
 72  2        call printx(a);

 73  2     end print;

 74  1     get$version: procedure addr;          /* returns current cp/m version # */

 75  2        return mon3(12,0);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 76  2     end get$version;


 77  1     conin: procedure byte;

 78  2         return mon2(6,0fdh);

 79  2     end conin;

 80  1     select: procedure(d);
 81  2         declare d byte;
 82  2         call mon1(14,d);
 83  2         end select;

 84  1     search$first: procedure(fcb) byte;
 85  2         declare fcb address;
 86  2         return mon2(17,fcb);
 87  2         end search$first;

 88  1     search$next: procedure byte;
 89  2         return mon2(18,0);
 90  2         end search$next;

 91  1     cselect: procedure byte;
           /* return current disk number */
 92  2         return mon2(25,0);
 93  2         end cselect;

 94  1     setdma: procedure(dma);
 95  2         declare dma address;
 96  2         call mon1(26,dma);
 97  2         end setdma;

 98  1     writeprot: procedure byte;          /* write protect the current disk */

 99  2         return mon2(28,0);

100  2     end writeprot;

101  1     getuser: procedure byte;            /* return current user number */

102  2         return mon2(32,0ffh);

103  2     end getuser;

104  1     return$errors: procedure(mode);     /* 0ff => return BDOS errors */
105  2         declare mode byte;

106  2           call mon1 (45,mode);

107  2     end return$errors;

108  1     setind: procedure(fcb) address;         /* SFA for current fcb */
109  2         dcl fcb addr;

110  2         call setdma(.passwd);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
111  2        return mon3(30,fcb);

112  2    end setind;

             /********** DISK PARAMETER BLOCK **********************/

113  1    declare
               dpba          address,
               dpb           based dpba structure(
                scptrK       address,
                blkshf       byte,
                blkmsK       byte,
                extmsK       byte,
                maxall       address,
                dirmax       address,
                dirblK       address,
                chksiz       address,
                offset       address,
                physhf       byte,
                phymsK       byte);


114  1    set$dpb: procedure;                  /* set disk parameter block values */

115  2        dpba = mon3(31,0);               /* base of dpb */

116  2    end set$dpb;

             /*****************************************************/

117  1    wrlbl: procedure(fcb) address;
118  2        declare fcb address;

119  2        call setdma(.passwd);       /* set dma=password */
120  2        return mon3(100,fcb);

121  2    end wrlbl;

122  1    getlbl: procedure(d) byte;
123  2        declare d byte;

124  2        return mon2(101,d);

125  2    end getlbl;

126  1    readxfcb: procedure(fcb) address;
127  2        declare fcb address;

128  2        call setdma(.passwd);       /* set dma=password */
129  2        return mon3(102,fcb);

130  2    end readxfcb;

131  1    wrxfcb: procedure(fcb) address;
132  2        declare fcb address;

133  2        call setdma(.passwd);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
134  2          return mon3(103,fcb);

135  2      end wrxfcb;


136  1      reset$drv: procedure(drv) byte;
137  2          dcl drv byte;

138  2          return mon2(37,reset$mask(drv));
139  2          end reset$drv;

140  1      parse: procedure(pfcb) address external;
141  2              declare pfcb address;

142  2      end parse;

143  1      delete: procedure(fcb) byte;
144  2              declare fcb     address;

145  2              return mon2(19,fcb);

146  2      end delete;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  GLOBAL DATA  * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

147  1          declare
                   fnam     literally '11',
                   ftyp     literally '9',
                   rofile   literally '9',     /* read/only file */
                   sysfile  literally '10',    /* system file */
                   archiv   literally '11',    /* archived  file */
                   attrb1   literally  '1',    /* attribute F1'  */
                   attrb2   literally  '2',    /* attribute F2'  */
                   attrb3   literally  '3',    /* attribute F3'  */
                   attrb4   literally  '4';    /* attribute F4'  */

148  1          declare
                       pwmask$on       literally '80h',
                       pwmask$off      literally '7fh',
                       acmask$on       literally '40h',
                       acmask$off      literally '0bfh',
                       upmask$on       literally '20h',
                       upmask$off      literally '0dfh',
                       crmask$on       literally '10h',
                       crmask$off      literally '0efh',
                       dlmask$on       literally '1h',
                       dlmask$off      literally '0feh';

149  1          declare
                   fcbp     address,
                   fcbv     based fcbp (32) byte,
                   fext     literally 'fcbv(12)';

150  1          declare
                   xfcb     (32) byte,
                   xfcbmode byte at (.xfcb(12));  /* password mode */

151  1          declare                          /* command buffer */
                   cmd (27) byte initial(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
                   passwd (17) byte;            /* password buffer */

152  1          declare
                       sfacmd   byte   initial(false), /* file attributes */
                       fileref  byte   initial(false), /* file reference  */
                       lblcmd   byte   initial(false), /* label attribute */
                       xfcbcmd  byte   initial(false), /* xfcb  attribute */
                       wild     byte   initial(false), /* file = a wildcard */
                       optdel   byte   initial(false), /* delimiter = option */
                       multi    byte   initial(false),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                    newpass  byte   initial(false),
                    passmsg  byte   initial(false),
                    NONBANK  byte   initial(false),
                    passmode byte,
                    password byte   initial(false); /* file has password */

153  1          declare                        /* parsing */
                   more  byte initial(true),   /* more to parse */
                   ibp        addr;            /* input buffer ptr */

154  1          declare
                   (sav$dcnt, sav$searcha)     addr,
                   sav$searchl  byte,
                   dirbuf (128) byte;          /* used for searches */

155  1          declare
                   cdisk     byte,             /* current disk */
                   ver       addr;             /* version checking */

156  1          declare
                   error$code addr;            /* for bdos returned
                                                  errors */

157  1          declare
                   parse$fn structure (
                       buff$adr  addr,
                       fcb$adr   addr),
                   last$buff$adr addr;         /* used for parsing */

158  1          declare
                    err$nofile(*)   byte data('Option requires a file reference',0),

                    err$driveonly(*) byte data('Option only for drives.',0),
                    errWASSPASS(*)  byte data('Assign passwords to input files.',0),

                    errASSPASS(*)   byte data('Assign a password to this file.',0),
                    errFORMAT(*)    byte data(
                    'Directory needs to be re-formatted for time/date stamps.',cr,
                    lf,'       Please see INITDIR.',0),
                    errNOPROT(*)    byte data('Protection not enabled for disk.',0),

                    errUNREC(*)     byte data('Unrecognized option.',0),
                    errNOMOD(*)     byte data
                                    ('There are no modifiers for this option.',0),
                    errUNRECM(*)    byte data
                                    ('Modifier missing or unrecognizable.',0),
                    errVALM(*)      byte data
                                    ('Not a valid modifier for this option.',0),
                    errOPTMOD(*)    byte data('This option needs a modifier.',0),
                    errBIGDEF(*)    byte data
                    ('Only first 8 characters of default password used.',0),
                    errBIGNAME(*)   byte data
                    ('Only first 11 characters of label name used.',0),
                    errBIGPASS(*)   byte data
                    ('Only first 8 characters of password used.',0),
                    errCRAC(*)      byte data
                           ('Cannot have both create and access time stamps.',0),
                    errSYSDIR(*)    byte data('Cannot set both sys and dir.',0),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        errRORW(*)      byte data('Cannot set RO and RW.',0),
                        errNOPT(*)      byte data('No options specified.',0),
                        errPAGE(*)      byte data('Page and nopage option selected.',
                                        '   Nopage in effect.',0),
                        errGLOBAL(*)    byte data
                                        ('Cannot set local options for file.',0),
                        errDrvProt(*)   byte data
                        ('Protection modifier is only ON/OFF for drives.',0),
                        errNBANK(*)     byte data
                        ('Password protection is not supported in NON-BANKED SYS.',0),
                        errVERS(*)      byte data('Requires CP/M 3 or higher.',0);

    $include (sopt.dcl)
=
159  1 =          declare
=                   opt$mod(19) structure(modifier(8) byte)
=                      data(1,1,1,0,0,0,0,0,                /* 0 access */
=                           1,1,1,0,0,0,0,0,                /* 1 archive */
=                           1,1,1,0,0,0,0,0,                /* 2 create */
=                           1,0,0,0,0,0,0,1,                /* 3 default */
=                           0,0,0,0,0,0,0,0,                /* 4 directory */
=                           1,1,1,0,0,0,0,0,                /* 5 f1 */
=                           1,1,1,0,0,0,0,0,
=                           1,1,1,0,0,0,0,0,
=                           1,1,1,0,0,0,0,0,
=                           1,0,0,0,0,0,0,1,                /* 9 name */
=                           1,0,0,0,0,0,0,1,                /* 10 password */
=                           1,1,1,1,1,1,1,0,                /* 11 protect */
=                           0,0,0,0,0,0,0,0,                /* 12 ro       */
=                           0,0,0,0,0,0,0,0,                /* 13 rw       */
=                           0,0,0,0,0,0,0,0,                /* 14 sys      */
=                           1,1,1,0,0,0,0,0,                /* 15 update */
=                           0,0,0,0,0,0,0,0,                /* 16 page */
=                           0,0,0,0,0,0,0,0),               /* 17 nopage */
=
=                   options(*) byte
=                          data('ACCESS0ARCHIVE0CREATE0DEFAULT0DIR0F10F20F30F40',
=                               'NAME0PASSWORD0PROTECTOR0ORW0SYS',
=                               '0UPDATE0PAGE0NOPAGE',0ffh),
=                   off$opt(20) byte data(0,7,15,22,30,34,37,40,43,46,51,60,68,71,
=                                         74,78,85,90,96),
=                   mods(*) byte
=                          data('OFF0ONREADWRITE0DELETE0NONE',0ffh),
=                   off$mods(7) byte data(0,4,7,12,18,25,29),
=
=                   end$list        byte data (0ffh),
=                   end$of$string   byte data(0),
=
=                   delimiters(*) byte data (0,'[]=, :;<>%\|''()/*!@&+-*?',0,0ffh),
=                   SPACE           byte data (5),    /* index in delim to space */
=                   RBRACKET        byte data(2),     /* ] in delim */
=                   ENDFF           byte data(25),
=                   EQUAL           byte data (3),
=                   LBRACKET        byte data (1),
=
=                   option$map(19)  byte,
=                   mods$map(19)    byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
        =
160   1 =       declare
        =               sfamsg          byte initial(false),
        =               drvmsg          byte initial(false),
        =               j               byte initial(0),
        =               string$ptr      address,
        =               defpass         address,
        =               labname         address,
        =               passname        address,
        =               lendef          byte,
        =               lenpass         byte,
        =               lenlab          byte,
        =               buf$ptr         address,
        =               index           byte,
        =               endbuf          byte,
        =               mindex          byte,
        =               delimiter       byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
        =  $ eject
        =

161  1          declare
                       scbpd   structure(
                        offs          byte,
                        set           byte,
                        value         address);

162  1          declare
                       line$page       byte,
                       line$out        byte,
                       savefcb(16)     byte,
                       save$dcnt       address,
                       save$searcha    address,
                       save$search1    address,
                       save$hash1      address,
                       save$hash2      address,
                       save$hash3      address,

                       COMbase         literally       '05dh',
                       page$off        literally       '01ch',
                       searcha$off     literally       '47h',
                       search1$off     literally       '49h',
                       dcnt$off        literally       '45h',
                       hash1$off       literally       '00h',
                       hash2$off       literally       '02h',
                       hash3$off       literally       '04h';

                                       /* get the scb word */
163  1     getscbword: procedure(off) address;
164  2          declare off     byte;

165  2          scbpd.offs = off;
166  2          scbpd.set = 0;
167  2          return mon3(49,.scbpd);

168  2     end getscbword;

169  1     setscb: procedure(off,value);
170  2          declare off     byte,
                       value   address;

171  2          scbpd.offs = off;
172  2          scbpd.set = 0feh;
173  2          scbpd.value = value;
174  2          call mon1(49,.scbpd);

175  2     end setscb;

176  1     getpage: procedure byte;

177  2          scbpd.offs = page$off;
178  2          scbpd.set = 0;
179  2          return mon2(49,.scbpd);

180  2     end getpage;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
            $eject

            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                              * * *  BASIC ROUTINES  * * *

             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                                    /* invalid command error */
181   1     perror: proc;

182   2        call print(.error$msg);
183   2        if ibp = 0 then call printx(parse$fn.buff$adr);
185   2        else call printx(last$buff$adr);

186   2        call printx(.(' ?',0));
187   2        call print(.invalid);
188   2        call terminate;
189   2     end perror;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                  /* parse the next lexical item in the command line
                     parse$fn must filled in with input parameters */
190   1     parser: procedure address;
191   2        declare p address;
192   2        declare c based p byte;

193   2           p = parse(.parse$fn);
194   2           if p = 0FFFFh then call perror;
196   2           else if p <> 0 then do;
198   3                  if c = '[' then optdel = true;
200   3                  else if c = ']' then optdel = false;
                           p = p + 1;
203   3           end;
204   2           else optdel = false;

205   2           return p;

206   2     end parser;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

207   1     fill:  proc(s,f,c);           /* fill string @ s for c bytes with f */
208   2        dcl s addr,
                  (f,c) byte,
                  a based s byte;

209   2           do while (c:=c-1)<>255;
210   3           a = f;
211   3           s = s+1;
212   3           end;
213   2        end fill;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
214  1     copy:  proc(s,d,c);          /* copy c bytes from s to d */
215  2        dcl (s,d) addr, c byte;
216  2        dcl a based s byte, b based d byte;

217  2           do while (c:=c-1)<>255;
218  3              b=a; s=s+1; d=d+1;
221  3           end;
222  2     end copy;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


223  1     ucase:  proc byte;           /* upper case character from console */
224  2        dcl c byte;

225  2        if (c:=conin) >= 'a' then
226  2           if c < '{' then
227  2              return(c-20h);
228  2        return c;
229  2     end ucase;

230  1     errprint: procedure(msg);
231  2           declare msg     address;

232  2           call print(.errormsg);
233  2           call printx(msg);
234  2           call crlf;

235  2     end errprint;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* get password and place in passwd */
236  1     getpasswd:  proc;
237  2        dcl (i,c) byte;

238  2           call print(.('Password ? ',0));

239  2     retry:
              call fill(.passwd,' ',8);
240  2           do i = 0 to 7;

241  3     nxtchr:
                     if (c:=ucase) >= ' ' then passwd(i)=c;
                     else
243  3                  if c = cr then go to exit;

                     if c = ctrlx then goto retry;
247  3                  if c = ctrlh then do;

249  4                        if i<1 then goto retry;
251  4                        else do;
252  5                              passwd(i:=i-1)=' ';
253  5                              goto nxtchr;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
254  5                          end;
255  4                  end;

256  3                  if c = ctrlc then call terminate;       /* end of program */
258  3          end;

259  2     exit:
                c = check$con$stat;            /* clear raw I/O mode */

260  2     end getpasswd;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                               /* print drive name */
261  1     printdrv: procedure;

262  2          call printchar(cdisk+'A');
263  2          call printchar(':');

264  2     end printdrv;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                                /* print file name */
265  1     printfn: procedure;
266  2          declare K byte;

267  2          call printdrv;

268  2          do K = 1 to fnam;
269  3                  if K = ftyp then call printchar('.');
271  3                  call printchar(fcbv(K) and 7fh);
272  3          end;

273  2     end printfn;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


274  1     bdos$error:   procedure;              /* error message routine */
275  2       declare
              code byte;

276  2          call print(.error$msg);
277  2          if (code:=high(error$code)) < 3 then do;
279  3                  call print(.error$msg);
280  3                  call printdrv;
281  3                  call printb;

282  3                  if code = 1 then call printx(.('Disk I/O',0));
284  3                  if code=2 then do;
286  4                      call printx(.('Drive ',0));
287  4                      call printx(.read$only);
288  4                  end;
289  3                  call terminate;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
290  3        end;

291  2           if code = 3 then call printx(.read$only);
293  2           if code = 4 then call printx(.('Invalid Drive.',0));
295  2           if code = 7 then call printx(.('Wrong Password',0));
297  2           if code = 9 then call printx(.('? in filespec.',0));

299  2     end bdos$error;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

300  1     set$search: procedure(dcnt);
301  2           declare dcnt    byte;

302  2           call setdma(.dirbuf);
303  2           dcnt = search$first(.('?'));

304  2     end set$search;

                                        /* get address of FCB in dirbuf */
305  1     set$up$file: procedure(dir$index);
306  2        dcl dir$index byte;

307  2        if dir$index <> 0ffh then do;
309  3           fcbp = shl(dir$index,5) + .dirbuf;
310  3           fcbv(0) = fcb(0);                /* set drive byte */
311  3        end;

312  2     end set$up$file;

313  1     getnext: procedure byte;
                                                 /* get the next fcb that matches fcb */

314  2           declare (dcnt,i)       byte;

315  2           xfcbcmd,sfacmd = false;


316  2           call setdma(.dirbuf);

                                                 /* restore saved search parameters */
317  2           call setscb(dcnt$off,save$dcnt);
318  2           call setscb(searcha$off,save$searcha);
319  2           call setscb(search1$off,save$search1);
320  2           call setscb(hash1$off,save$hash1);
321  2           call setscb(hash2$off,save$hash2);
322  2           call setscb(hash3$off,save$hash3);
323  2           call copy(.savefcb,save$searcha,16);

324  2           if (dcnt != search$next) = 0ffh then return(false);
326  2           call set$up$file(dcnt);
327  2           return(true);

328  2     end getnext;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* print boolean option value */
329   1     pbool: procedure(value);
330   2           declare
                     value byte;

331   2           call printx(.option$set);
332   2           if value then call printx(.('ON',0));
334   2           else call printx(.('OFF',0));

335   2     end pbool;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


            /*************************************************************

                        F I L E   A T T R I B U T E S

            *************************************************************/



336   1     printatt: procedure;          /* print attribute set */

337   2       attribute: procedure(i) byte; /* test if attribute fcbv(i) is on */
338   3            declare i byte;

339   3            if rol(fcbv(i),1) then return true;
341   3            return false;
342   3       end attribute;

                                          /* display attributes: sys,ro,a,f1-f4 */

343   2            call printx(.set$to);
344   2            if attribute(sysfile) then call printx(.('system (SYS)',0));
346   2            else call printx(.('directory (DIR)',0));

347   2            call printx(.(', ',0));
348   2            if attribute(rofile) then do;
350   3                   call printx(.read$only);
351   3                   call printx(.ro);
352   3            end;
353   2            else call printx(.read$write);

354   2            call printchar(tab);
355   2            if attribute(archiv) then call printchar('A');
357   2            if attribute( attrb1 ) then call printchar('1');
359   2            if attribute( attrb2 ) then call printchar('2');
361   2            if attribute( attrb3 ) then call printchar('3');
363   2            if attribute( attrb4 ) then call printchar('4');

365   2     end print$att;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                        /* read current file attributes */
366  1      rd$attributes: procedure;

367  2          if not sfcmd then               /* have read the FCB yet? */
368  2              if not wild then do;
370  3                  call setdma(.dirbuf);
371  3                  call set$up$file(search$first(.fcb));
372  3              end;

373  2      end rd$attributes;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


            /*************************************************************

                        D R I V E   A T T R I B U T E S

            *************************************************************/


374  1      setdrvstatus: procedure(func);      /* set drive attributes */

375  2              declare
                            code    byte,
                            func    byte;

                                                        /* set the drive */
376  2          if func = opt$ro then code = writeprot;   /* read only */
               else
378  2              code = reset$drv(cdisk);            /* read/write */

                                                        /* display */
379  2          if code <> 0ffh then do;
381  3              call print(.('Drive ',0));
382  3              call printdrv;
383  3              call printb;
384  3              call printx(.set$to);
385  3              if func = opt$ro then do;
387  4                  call printx(.read$only);
388  4                  call printx(.ro);
389  4              end;
                    else
390  3                  call printx(.read$write);
391  3          end;

392  2      end setdrvstatus;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


            /*************************************************************

                        L A B E L   A T T R I B U T E S

            *************************************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                        /* read the directory label before
                                           writing the label to preserve the
                                           name, type, and stamps */
393  1      readlabel: procedure;
394  2          dcl (mode, dcnt) byte;

            /*-------------------------------------------------------------*/
395  2      readlbl: proc;
396  3          dcl d byte data('?');

397  3          call setdma(.dirbuf);
398  3          dcnt = search$first(.d);          /* position to first dcnt in dir */
399  3          do while dcnt <> 0ffh;            /* read entire directory */
                                                 /* is the user# a label = 20h */
400  4              if dirbuf(ror(dcnt,3) and 110$0000b)=20H then return;
402  4              dcnt = search$next;
403  4          end;

404  3      end readlbl;

            /*-------------------------------------------------------------*/

405  2          if lblcmd then return;

407  2          mode = getlbl(cdisk);             /* get the dir label data byte */
408  2          password = false;
409  2          if mode > 0 then do;              /* if ok then ...*/
411  3              call readlbl;                 /* get label */
412  3              fcbp = shl(dcnt,5) + .dirbuf;
413  3              fext = fext and 11110000b;    /* turn off set passwd */
414  3              if fcbv(16) <> ' ' then
415  3                  if fcbv(16) <> 0 then
416  3                      password = true;
417  3          end;

418  2          else do;                          /* no dir label */
419  3              fcbp = .fcb;
420  3              call copy(.label$name,.fcb(1),length(label$name));
421  3          end;

            /*    if password then call getpasswd;*/   /* does the user have the password*/
422  2          lblcmd = true;

423  2      end readlabel;


            /*****************************************************************************/


424  1      put$file: procedure;                  /* display the file or xfcb */

425  2          call crlf;
426  2          call printfn;
427  2          call printb;
428  2          call printb;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
429  2     end put$file;

           /*************************************************************

                        S F C B    A T T R I B U T E S

           *************************************************************/


430  1     set$up$xfcb: procedure;                /* read xfcb into xfcb buffer */

431  2             if not xfcbcmd then do;
433  3                     xfcbcmd = true;
434  3                     call copy(.fcbv,.xfcb,12);
435  3                     password,passmode = 0;

436  3                     if low(errorcode := readxfcb(.xfcb)) = 0ffh then do;
438  4                             if high(errorcode) <> 0 then call bdos$error;
440  4                             else do;
441  5                                     call errprint(.not$found);
442  5                                     call put$file;
443  5                             end;
444  4                             return;
445  4                     end;

446  3                     passmode = xfcb(12);
447  3                     if passmode <> 0 then password = true;  /* must have a pass if
                                                                     mode <> NONE */
449  3             end;

450  2     end set$up$xfcb;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


           /*************************************************************

                   PASSWORD  AND  PASSWORD  MODE  ROUTINES

           *************************************************************/


451  1     defaultpass: procedure;

452  2             if NONBANK then do;
454  3                     call errprint(.errNBANK);
455  3                     return;
456  3             end;

457  2             call fill(.passwd(0),' ',8);
458  2             call copy(defpass,.passwd(0),lendef);
459  2             call mon1(106,.passwd);
460  2             call print(.('Default password = ',0));
461  2             passwd(8) = 0;
462  2             call printx(.passwd);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
463  2     end defaultpass;


464  1     set$password: procedure;

465  2           if fileref then do;

467  3                 if NONBANK then do;
469  4                       call errprint(.errNBANK);
470  4                       return;
471  4                 end;

472  3                 call set$up$xfcb;
473  3                 passmode = passmode or 1;       /* turn on password bit */
474  3           end;
475  2           else do;
476  3                 call readlabel;
477  3                 fext = fext or 1;
478  3           end;

479  2           call fill(.passwd(8),' ',8);          /* clear passwd */

480  2           if lenpass = 0 then do;
482  3                 passmode = 1;
483  3                 return;
484  3           end;

485  2           newpass = true;
486  2           call copy(passname,.passwd(8),lenpass);  /* copy it to fcb */

487  2     end set$password;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


           /*************************************************************

                       LABEL  ATTRIBUTE  ROUTINES

           *************************************************************/

488  1     lname: procedure;                       /* sets the label name */
489  2           declare i      byte,
                      ln     based labname (1) byte;

490  2           if drvmsg then return;

492  2           if fileref then do;
494  3              call errprint(.err$driveonly);
495  3              drvmsg = true;
496  3              return;
497  3           end;

498  2           call readlabel;

499  2           call fill(.fcbv(1),' ',11);           /* clear name */
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
500  2          if lenlab > 0 then do;
502  3             do i = 0 to lenlab-1;
503  4                if ln(i) = PERIOD then do;
505  5                      call copy(labname,.fcbv(1),i);
506  5                      call copy(labname+i+1,.fcbv(9),3);
507  5                      return;
508  5                end;
509  4             end;

510  3             call copy(labname,.fcbv(1),lenlab);   /* copy label name */

511  3          end;

512  2     end lname;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


513  1     set$extent: procedure(function,maskon,maskoff);
514  2          declare
                      function       byte,
                      maskon         byte,
                      maskoff        byte;

515  2          if drvmsg then return;

517  2          if fileref then do;
519  3                drvmsg = true;
520  3                call errprint(.err$driveonly);
521  3                return;
522  3          end;

523  2          call readlabel;
524  2          if mods$map(function) then fext = fext or maskon;   /* turn stamp on */
526  2          else fext = fext and maskoff;                       /* turn stamp off */

527  2          return;

528  2     end set$extent;


529  1     protect: procedure;                    /* set drive protection mode */
530  2          declare pmode   byte;

531  2          if fileref then  do;
533  3                call set$up$xfcb;
534  3                pmode = mods$map(opt$prot);

535  3                if pmode = 2 then passmode = 80h;       /* read only */
                      else
537  3                if pmode = 3 then passmode = 40h;       /* write,read */
                      else
539  3                if pmode = 4 then passmode = 20h;       /* r,w,delete */
541  3                else do ;
542  4                      passmode = 1;                     /* turn off protection*/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
543  4                        call fill(.passwd(8),' ',8);
544  4                  end;
545  3                  if newpass then passmode = passmode or 1;
547  3            end;
548  2            else do;

549  3                  if NONBANK then do;
551  4                        call errprint(.errNBANK);
552  4                        return;
553  4                  end;

554  3                  pmode = mods$map(opt$prot);
555  3                  if pmode > 1 then do;
557  4                        call errprint(.errDrvProt);
558  4                        return;
559  4                  end;

560  3                  call set$extent(opt$prot,pwmask$on,pwmask$off);
561  3                  call fill(.fcbv(16),' ',8);            /* erase password */
562  3            end;

563  2      end protect;

            /*-------------------------------------------------------------*/

                                              /* set attribute bits:
                                                     f1 --> f4 flags
                                                     t1 --> t3 flags or
                                                        RO
                                                        SYS
                                                        Archive     */

564  1      setatt: procedure(func,bytes);
565  2            declare func    byte,
                        bytes   byte;


566  2            if sfamsg then return;                 /* printed msg before? */
568  2            if not fileref then do;
570  3                  sfamsg = true;
571  3                  call errprint(.err$nofile);
572  3                  return;
573  3            end;

574  2            if mods$map(func) then fcbv(bytes) = fcbv(bytes) or 80h;
576  2            else fcbv(bytes) = fcbv(bytes) and 7fh;

577  2            sfacmd = true;
578  2      end setatt;

            /*************************************************************

                     S H O W   L A B E L   &   X F C B

            *************************************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
579  1      show$passwd: procedure;                /* display the new password */

580  2          call printx(.('Password = ',0));
581  2          passwd(16) = 0;
582  2          call printx(.passwd(8));

583  2      end show$passwd;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


584  1      dcl label1 (*) byte data (
            'Directory      Passwds  Stamp    Stamp    Stamp',cr,lf,
            'Label          Reqd     Create   Access   Update',cr,lf,
            '-------------  -------  -------  -------  -------',cr,lf,0);

585  1      showlbl: procedure;                    /* show the label options */
586  2          declare (make,access) byte;

587  2          call print(.('Label for drive ',0));
588  2          call printdrv;
589  2          call crlf;
590  2          call print(.label1);
591  2          call printfn;

592  2          if (fext and 80h) = 80h then       /* PASSWORDS REQUIRED */
593  2              call printx(.on);
                else
594  2              call printx(.off);

595  2          access = (fext and 40h) = 40h;     /* STAMP CREATE */
596  2          if (fext and 10h) = 10h then
597  2              call printx(.on);
                else
598  2              call printx(.off);

599  2          if access then                     /* STAMP ACCESS */
600  2              call printx(.on);
                else
601  2              call printx(.off);

602  2          if (fext and 20h) = 20h then       /* STAMP UPDATE */
603  2              call printx(.on);
                else
604  2              call printx(.off);

605  2          call crlf;
606  2          if fext then do;
608  3              call crlf;
609  3              call show$passwd;
610  3          end;

611  2      end showlbl;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
612  1    show$xfcb: procedure;                    /* display xfcb attributes */

613  2         call printx(.('Protection = ',0));

614  2         if (passmode and 80h) = 80h then call printx(.readmode);
               else
616  2         if (passmode and 40h) = 40h then call printx(.writemode);
               else
618  2         if (passmode and 20h) = 20h then call printx(.deletemode);
               else
620  2         if (not passmode) or (passwd(8) = ' ') then call printx(.nopasswd);
               else
622  2             call printx(.readmode);

623  2         if passmode then do;                 /* lsb on */
625  3                call printx(.comma);
626  3                call show$passwd;
627  3         end;

628  2    end show$xfcb;

          /*************************************************************

                WRITE  XFCB, LABEL  AND  FILE  ATTRIBUTES

          *************************************************************/

629  1    pass$check: procedure(which) byte;
630  2         declare which   byte;
                                        /* did we fail because of password?
                                           if so, then get it and re-try.
                                                which = 1 <-- put$attribute
                                                        2 <-- write$label
                                                        3 <-- write$xfcb */
631  2         if high(error$code) = 7 then do;
633  3                call crlf;
634  3                if which <> 2 then call put$file;
636  3                else call print(.dirlabel);
637  3                call getpasswd;
638  3                if fileref then call crlf;
                                        /* put attributes ? */
640  3                if which = 1 then error$code = setind(fcbp);
                      else              /* write label ? */
642  3                if which = 2 then error$code = wrlbl(fcbp);
                      else              /* update xfcb */
644  3                       error$code = wrxfcb(.xfcb);

645  3                if high(error$code) <> 0 then do;
647  4                       call bdos$error;
648  4                       if which = 2 then call print(.dirlabel);
650  4                       else call put$file;
651  4                       return(false);
652  4                end;
653  3         end;

654  2         return(true);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
655  2      end pass$check;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

656  1      put$attributes: procedure;              /* write file attributes */

657  2          error$code = setind(fcbp);

658  2          if low(error$code) = 0ffh then
659  2              if high(error$code) <> 0 then do;
661  3                      if not pass$check(1) then return;
663  3                      if high(error$code) <> 0 then do;
665  4                              call bdos$error;
666  4                              call put$file;
667  4                              return;
668  4                      end;
669  3              end;
670  2              else do;
671  3                      call errprint(.not$found);
672  3                      call put$file;
673  3              end;

674  2          if low(error$code) <> 0ffh then
675  2              if fext <= dpb.extmsk then do;
677  3                  call put$file;
678  3                  call print$att;
679  3              end;

680  2      end put$attributes;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

681  1      write$label: procedure;                 /* write new label */

682  2          error$code = wrlbl(fcbp);

683  2          if low(error$code) = 0ffh then
684  2              if high(error$code) <> 0 then do;
686  3                      if not pass$check(2) then return;
688  3                      if high(error$code) <> 0 then do;
690  4                              call bdos$error;
691  4                              call print(.dirlabel);
692  4                              return;
693  4                      end;
694  3                      call crlf;
695  3              end;
696  2              else do;
697  3                      call errprint(.errFORMAT);
698  3                      return;
699  3              end;

700  2              call showlbl;

701  2      end write$label;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

 702   1        write$xfcb: procedure;                  /* write out new xfcb */

 703   2                if passmode > 1 then do;
 705   3                        if password then go to wr0;
 707   3                        if newpass then go to wr0;

 709   3                                if passmsg then return;

 711   3                                if wild then
 712   3                                   call errprint(.errWASSPASS);
 713   3                                else do;
 714   4                                   call errprint(.errASSPASS);
 715   4                                   call put$file;
 716   4                                end;

 717   3                                passmsg = true;
 718   3                                return;
 719   3                end;

 720   2        wr0:    if passmode = 1 then
 721   2                   if newpass then passmode = passmode or 80h;  /* read mode = def */

 723   2                xfcbmode = passmode;
 724   2                error$code = wrxfcb(.xfcb);

 725   2                if low(error$code) = 0ffh then
 726   2                   if high(error$code) <> 0 then do;
 728   3                        if not pass$check(3) then return;
 730   3                        if high(error$code) <> 0 then do;
 732   4                                call bdos$error;
 733   4                                call put$file;
 734   4                                return;
 735   4                        end;
 736   3                   end;
 737   2                   else do;
 738   3                        call errprint(.not$found);
 739   3                        call print(.('       or protection not enabled for disk.',0));
 740   3                        return;
 741   3                   end;

 742   2                if passmode = 1 then do;                /* delete xfcb */
 744   3        wr1:            xfcb(5) = xfcb(5) or 80h;
 745   3                        error$code = delete(.xfcb);     /* no need to check for error*/
 746   3                end;                                    /* previous write-> failed!*/


 747   2                call put$file;
 748   2                call show$xfcb;                 /* errcode is good if we are here */

 749   2        end write$xfcb;
```

CP/M - PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
            /*************************************************************

                        C O M M A N D   P R O C E S S I N G

            *************************************************************/


750   1     setdisk: procedure;           /* select the disk specified in cmd line */

751   2         if cmd(0) <> 0 then do;
753   3             cdisk = cmd(0)-1;
754   3             call select(cdisk);
755   3             call set$dpb;
756   3             end;

757   2     end setdisk;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


758   1     wildcard: procedure byte;             /* test if the file is a wildcard */
759   2         declare
                i byte;

760   2             do i=1 to fnam;
761   3                   if fcb(i) = '?' then return true;
763   3             end;
764   2             return false;
765   2     end wildcard;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


766   1     setup$fcb: procedure;         /* set up the next file or drive reference */
767   2             declare dcnt    byte;

768   2             call setdisk;
769   2             call copy(.cmd,.fcb,12);      /* name */
770   2             call copy(.cmd(16),.passwd,8); /* password */

771   2             if fcb(1) <> ' ' or fcb(ftyp) <> ' ' then do;
773   3                     fileref = true;
774   3                     call setdma(.dirbuf);
775   3                     if (dcnt := search$first(.fcb)) = 0ffh then do;
777   4                           fcbp = .fcb;
778   4                           call errprint(.not$found);
779   4                           call put$file;
780   4                           call terminate;
781   4                     end;
782   3                     call set$up$file(dcnt);
783   3             end;
784   2             else fileref = false;

785   2     end setup$fcb;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
$include (sopt.inc)
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
        =   $eject
786   1 =   check$choice: procedure(index,mindex) byte;
        =                                       /* does this modifier go with this
        =                                          option? */
787   2 =         declare
        =                 index   byte,
        =                 mindex  byte;
        =
788   2 =         return(opt$mod(index).modifier(mindex));
        =
789   2 =   end check$choice;
        =
        =   /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        =
        =
        =                    * * *  Option scanner  * * *
        =
        =
        =    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
        =
        =
790   1 =   separator: procedure(character) byte;
        =
        =                                       /* determines if character is a
        =                                          delimiter and which one */
791   2 =         declare K       byte,
        =                 character       byte;
        =
792   2 =         K = 1;
793   2 =   loop:   if delimiters(K) = end$list then return(0);
795   2 =         if delimiters(K) = character then return(K);    /* null = 25 */
797   2 =                 K = K + 1;
798   2 =                 go to loop;
        =
799   2 =   end separator;
        =
800   1 =   opt$scanner:    procedure(list$ptr,off$ptr) byte;
        =
        =               /* list$ptr        -       pointer to list of known strings
        =                  off$ptr         -       pointer to offsets into known string
        =                                          list
        =                  buf$ptr         -       pointer to input string
        =
        =                  Scans the known string list for an occurrance of the input
        =                  string.  If the input string is not found in the known list
        =                  then return(0). Otherwise, return the index of the known string
        =                  that matches the input.
        =
        =                  1. Find the known string that matches the input string on the
        =                     first letter.
        =
        =                          do i = 1 to #known_strings
        =                                  if known_string(i,1) = input(1) then do
        =
        =                                     if length(known_string(i)) < end_of_input
        =                                        then return(0)
        =
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                    do j = 2 to end_of_input

                                        if Known_string(i,j) <> input(j) then
                                                go to again
                                    end

                                    go to 2
                                end
                        again: end

                                return (0)              !no matchs

                        2. Test to see if the input string does not match another Known
                           string.  This may happen if the input string is not a
                           unique sub-string of the Known string, ie., DI is a
                           sub-string of DIRECTORY and DISK.

                                index = i

                                do i = index+1 to #Known_strings
                                        do j = 1 to end of input

                                                if Known_string(i,j) <> input(j) then
                                                        go to next
                                        end

                                                return(0)       !not unique
                        next:   end;

                                return(index)           !unique substring

                                        P.Balma   10/82  */

801   2         declare
                        buff            based buf$ptr (1) byte,
                        off$ptr         address,
                        list$ptr        address;

802   2         declare
                        i               byte,
                        j               byte,
                        list            based list$ptr (1) byte,
                        offsets         based off$ptr (1) byte,
                        wrd$pos         byte,
                        character       byte,
                        letter$in$word  byte,
                        found$first     byte,
                        start           byte,
                        index           byte,
                        save$index      byte,
                        (len$new,len$found)     byte,
                        valid           byte;

        /*****************************************************************************/
        /*                      internal subroutines                                 */
        /*****************************************************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
803  2  =  check$in$list: procedure;
        =                              /* find known string that has a match with
        =                                 input on the first character.  Set index
        =                                 = invalid if none found.   */
        =
804  3  =       declare i      byte;
        =
805  3  =       i = start;
806  3  =       wrd$pos = offsets(i);
807  3  =       do while list(wrd$pos) <> end$list;
808  4  =             i = i + 1;
809  4  =             index = i;
810  4  =             if list(wrd$pos) = character then return;
812  4  =             wrd$pos = offsets(i);
813  4  =       end;
        =                        /* could not find character */
814  3  =       index = 0;
815  3  =       return;
816  3  =  end check$in$list;
        =
817  2  =  setup:  procedure;
818  3  =       character = buff(0);
819  3  =       call check$in$list;
820  3  =       letter$in$word = wrd$pos;
        =                        /* even though no match may have occurred, position
        =                           to next input character.  */
821  3  =       i = 1;
822  3  =       character = buff(1);
823  3  =  end setup;
        =
824  2  =  test$letter:   procedure;
        =                        /* test each letter in input and known string */
        =
825  3  =       letter$in$word = letter$in$word + 1;
        =
        =                                    /* too many chars input? 0 means
        =                                       past end of known string */
826  3  =       if list(letter$in$word) = end$of$string then valid = false;
        =       else
828  3  =       if list(letter$in$word) <> character then valid = false;
        =
        =       i = i + 1;
831  3  =       character = buff(i);
        =
832  3  =  end test$letter;
        =
833  2  =  skip:   procedure;
        =                                    /* scan past the offending string;
        =                                       position buf$ptr to next string...
        =                                       skip entire offending string;
        =                                       ie., falseopt=mod, [note: comma or
        =                                       space is considered to be group
        =                                       delimiter] */
834  3  =       character = buff(i);
835  3  =       delimiter = separator(character);
836  3  =       do while ((delimiter <> 2) and (delimiter <> 4) and (delimiter <> 5)
        =                 and (delimiter <> 25));
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
837  4  =                 i = i + 1;
838  4  =                 character = buff(i);
839  4  =                 delimiter = separator(character);
840  4  =           end;
841  3  =           endbuf = i;
842  3  =           buf$ptr = buf$ptr + endbuf + 1;
843  3  =           return;
844  3  =   end skip;
        =
845  2  =   eat$blanks: procedure;
        =
846  3  =           declare charac  based buf$ptr byte;
        =
847  3  =           do while(delimiter <> separator(charac)) = SPACE;
848  4  =                 buf$ptr = buf$ptr + 1;
849  4  =           end;
        =
850  3  =   end eat$blanks;
        =
        =   /*************************************************************************/
        =   /*                        end of internals                               */
        =   /*************************************************************************/
        =
        =
        =                                       /* start of procedure */
851  2  =           call eat$blanks;
852  2  =           start = 0;
853  2  =           call setup;
        =
        =                                       /* match each character with the option
        =                                          for as many chars as input
        =                                          Please note that due to the array
        =                                          indices being relative to 0 and the
        =                                          use of index both as a validity flag
        =                                          and as a index into the option/mods
        =                                          list, index is forced to be +1 as an
        =                                          index into array and 0 as a flag*/
        =
854  2  =           do while index <> 0;
855  3  =                 start = index;
856  3  =                 delimiter = separator(character);
        =
        =                                       /* check up to input delimiter */
        =
857  3  =                 valid = true;          /* test$letter resets this */
858  3  =                 do while delimiter = 0;
859  4  =                       call test$letter;
860  4  =                       if not valid then go to exit1;
862  4  =                       delimiter = separator(character);
863  4  =                 end;
        =
864  3  =                 go to good;
        =
        =                                       /* input <> this known string;
        =                                          get next known string that
        =                                          matches */
865  3  =   exit1:           call setup;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
866  3  =          end;                              /* fell through from above, did
     =                                                 not find a good match*/
     =                                              /* skip over string & return*/
867  2  =          endbuf = i;
868  2  =          call skip;
869  2  =          return(0);
     =
     =                                              /* is it a unique match in options
     =                                                 list? */
870  2  =  good:   endbuf = i;
871  2  =          len$found = endbuf;
872  2  =          save$index = index;
873  2  =          valid = false;
874  2  =  next$opt:
     =                    start = index;
875  2  =                    call setup;
876  2  =                    if index = 0 then go to finished;
     =                                              /* look at other options and check
     =                                                 uniqueness */
     =
878  2  =                    len$new = offsets(index + 1) - offsets(index) - 1;
879  2  =                    if len$new = len$found then do;
881  3  =                           valid = true;
882  3  =                           do j = 1 to len$found;
883  4  =                                  call test$letter;
884  4  =                                  if not valid then go to next$opt;
886  4  =                           end;
887  3  =                    end;
888  2  =                    else go to nextopt;
     =                                              /* fell through...found another valid
     =                                                 match --> ambiguous reference */
889  2  =          call skip;                      /* skip input field to next delimiter*/
890  2  =          return(0);
     =
891  2  =  finished:                               /* unambiguous reference */
     =          buf$ptr = buf$ptr + endbuf;
892  2  =          call eat$blanks;
893  2  =          if delimiter <> 0 then buf$ptr = buf$ptr + 1;
895  2  =          else delimiter = SPACE;
     =
896  2  =          return(save$index);
     =
897  2  =  end opt$scanner;
     =
898  1  =  error$prt:     procedure;
899  2  =          declare i       byte,
     =                  t       address,
     =                  char    based t byte;
     =
900  2  =          t = buf$ptr - endbuf - 1;
901  2  =          do i = 1 to endbuf;
902  3  =                 call printchar(char);
903  3  =                 t = t + 1;
904  3  =          end;
     =
905  2  =  end error$prt;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

=

```
906  1     parse$options: procedure;

907  2          declare
                      charac          based buf$ptr byte,
                      1               byte;

908  2          delimiter = 1;
909  2          index = 0;
910  2          mindex = 0;

911  2     loop:
                if delimiter = 0 then return;
913  2          if delimiter = RBRACKET then return;
915  2          if delimiter = ENDFF then return;

                                        /* get the index into list */
917  2          if (index := opt$scanner(.options,.off$opt)) = 0 then go to error1;

                                        /* if we have more to parse,
                                           check for valid modifiers */
919  2          if (delimiter <> RBRACKET and delimiter <> ENDFF) then do;

                                        /* is this a mod delimiter?
                                           test for equal sign. */
921  3               if delimiter = EQUAL then do;
                                        /* does option have a modifier?*/

923  4                    if not opt$mod(index-1).modifier(0) then go to error2;

                                        /* is this a string modifier, ie.,
                                           password,default,name option */

925  4                    if not opt$mod(index-1).modifier(7) then do;

927  5                         if (mindex := opt$scanner(.mods,.off$mods)) = 0
                                  then go to error3;

                                        /* invalid option-modifier pair */

929  5                         if not opt$mod(index-1).modifier(mindex) then
930  5                              go to error4;

931  5                    end;          /* ends getting non-string mod */

932  4                    else do;
                                        /* get string */
933  5                         string$ptr = buf$ptr;
934  5                         mindex = 8;
935  5                         delimiter = 0;
936  5                         l = 0;
937  5                         do while delimiter = 0;
938  6                              delimiter = separator(charac);
939  6                              buf$ptr = buf$ptr + 1;
940  6                              l = l + 1;
941  6                         end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
942  5                                  if delimiter = SPACE then do;
944  6                                        delimiter = separator(charac);
945  6                                        buf$ptr = buf$ptr + 1;
946  6                                  end;

947  5                                  l = l - 1;
948  5                                  if l > 0 then do;
950  6                                     if (index -1) = opt$default then do;
952  7                                        defpass = string$ptr;
953  7                                        if (lendef := l) > 8 then do;
955  8                                           call errprint(.errBIGDEF);
956  8                                           lendef = 8;
957  8                                        end;
958  7                                     end;
                                           else
959  6                                     if (index -1) = opt$name then do;
961  7                                        labname = string$ptr;
962  7                                        if (lenlab := l) > 11 then do;
964  8                                           lenlab = 11;
965  8                                           call errprint(.errBIGNAME);
966  8                                        end;
967  7                                     end;
968  6                                     else do;
969  7                                        passname = string$ptr;
970  7                                        if (lenpass := l) > 8 then do;
972  8                                           call errprint(.errBIGPASS);
973  8                                           lenpass= 8;
974  8                                        end;
975  7                                     end;
976  6                                  end;
977  5                            end;
978  4                     end;           /* ends mod delimiter? */
979  3              end;                  /* ends last delimiter */

                                                 /* option without modifier...
                                                    index must be > 0 */
980  2           if mindex = 0 and opt$mod(index-1).modifier(0) then go to error5;

982  2           option$map(index - 1) = true;
983  2           if mindex > 0 then mods$map(index - 1) = mindex - 1;

985  2           go to loop;                   /* skip error routine */

986  2     error1: call errprint(.errUNREC);
987  2             go to optprt;
988  2     error2: call errprint(.errNOMOD);
989  2             go to optprt;
990  2     error3: call errprint(.errUNRECM);
991  2             go to modprt;
992  2     error4: call errprint(.errVALM);
993  2             go to modprt;
994  2     error5: call errprint(.errOPTMOD);
995  2             go to optprt;
996  2     modprt: call print(.('Modifier: ',0));
997  2             go to errprt;
998  2     optprt: call print(.('Option: ',0));
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 999  2     errprt: call error$prt;

1000  2             go to loop;

1001  2     end parse$options;

1002  1     do$options:     procedure;
1003  2             declare dump    byte;

1004  2             if option$map(opt$archive) then
1005  2                     call setatt(opt$archive,archiv);

1006  2             if option$map(opt$f1) then call setatt(opt$f1,attrb1);
1008  2             if option$map(opt$f2) then call setatt(opt$f2,attrb2);
1010  2             if option$map(opt$f3) then call setatt(opt$f3,attrb3);
1012  2             if option$map(opt$f4) then call setatt(opt$f4,attrb4);

1014  2             if option$map(opt$name) then call lname;        /*Dir name*/
1016  2             if option$map(opt$pass) then call set$password;
1018  2             if option$map(opt$prot) then call protect;
1020  2             if option$map(opt$default) then call defaultpass;

1022  2             if option$map(opt$access) and option$map(opt$create) then do;
1024  3                     if mods$map(opt$access) and mods$map(opt$create) then do;
1026  4                             if fileref then call errprint(.err$driveonly);
1028  4                             call errprint(.errCRAC);
1029  4                             call crlf;
1030  4                             go to do1;
1031  4                     end;
1032  3             end;

1033  2             if option$map(opt$access) then do;
1035  3                     if mods$map(opt$access) then do;        /* turn off create */
1037  4                             mods$map(opt$create) = 0;
1038  4                             call set$extent(opt$create,crmask$on,crmask$off);
1039  4                     end;
1040  3                     call set$extent(opt$access,acmask$on,acmask$off);
1041  3             end;
1042  2             if option$map(opt$create) then do;
1044  3                     if mods$map(opt$create) then do;        /* turn off access */
1046  4                             mods$map(opt$access) = 0;
1047  4                             call set$extent(opt$access,acmask$on,acmask$off);
1048  4                     end;
1049  3                     call set$extent(opt$create,crmask$on,crmask$off);
1050  3             end;

                                        /* Note that sys and dir do NOT have
                                           modifiers; thus the option scanner
                                           did not fill in the modifier map,
                                           which setatt looks at to turn things
                                           on/off.  So we have to set the mod
                                           map here.  applies to archive too */

1051  2     do1:    if option$map(opt$dir) and option$map(opt$sys) then do;
1053  3                     if not fileref then call errprint(.err$nofile);
1055  3                     call errprint(.errSYSDIR);
1056  3                     call crlf;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1057  3           end;
1058  2           else do;
1059  3                 if option$map(opt$dir) then
                                                  /* do not turn sys on */
1060  3                       call setatt(opt$sys,sysfile);

1061  3                 else if option$map(opt$sys) then do;
1063  4                       mods$map(opt$sys) = true;
1064  4                       call setatt(opt$sys,sysfile);
1065  4                 end;
                  end;

1067  2           if option$map(opt$update) then
1068  2                 call set$extent(opt$update,upmask$on,upmask$off);

1069  2           if option$map(opt$ro) and option$map(opt$rw) then do;
1071  3                 call errprint(.errRORW);
1072  3                 call crlf;
1073  3           end;
1074  2           else do;
1075  3                 if option$map(opt$ro) then
1076  3                    if fileref then do;
1078  4                       mods$map(opt$ro) = 1;
1079  4                       call setatt(opt$ro,rofile);
1080  4                    end;
1081  3                    else call setdrvstatus(opt$ro);
                      else
1082  3                 if option$map(opt$rw) then
1083  3                    if fileref then do;
                                                  /* turn ro off */
1085  4                       mods$map(opt$ro) = 0;
1086  4                       call setatt(opt$ro,rofile);
1087  4                    end;
1088  3                    else call setdrvstatus(opt$rw);
            end;
1090  2     end do$options;

1091  1     save: procedure;

                                    /* save search parameters for later wild
                                       card processing */

1092  2           save$dcnt = getscbword(dcnt$off);
1093  2           save$searcha = getscbword(searcha$off);
1094  2           save$searchl = getscbword(searchl$off);
1095  2           save$hash1 = getscbword(hash1$off);
1096  2           save$hash2 = getscbword(hash2$off);
1097  2           save$hash3 = getscbword(hash3$off);

1098  2     end save;


1099  1     savewild: procedure;

                                    /* save wildcard name for later processing */
1100  2           if (wild != wildcard) then call copy(.cmd,.savefcb,12);
1102  2           call setup$fcb;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1103  2     end savewild;


1104  1     getfilename: procedure(buffadd);
1105  2           declare buffadd        address;

1106  2           parse$fn.buff$adr = buffadd;
1107  2           last$buff$adr = buffadd;       /* used by perror routine */
1108  2           parse$fn.fcb$adr = .cmd;
1109  2           ibp = parser;                   /* parse file name */

1110  2     end getfilename;

1111  1     getfname: procedure;

1112  2           call getfilename(bufptr);

1113  2           if optdel then do;                    /* no local options */
1115  3                 call errprint(.errGLOBAL);
1116  3                 cmd(12) = 0;
1117  3                 call print(.('FILE: ',0));
1118  3                 call printx(.cmd(1));
1119  3                 call terminate;
1120  3           end;
                                                      /* F152 returns ^= 0 if
                                                         another file name
                                                         follows in buffer */
1121  2           if ibp <> 0 then multi = true;
1123  2           else multi = false;

1124  2           call copy(.cmd,.fcb,16);              /* copy file name to
                                                         default buffer..*/
1125  2           call savewild;

1126  2     end getfname;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
            $eject
            /*********************************************************

                        M A I N  P R O G R A M

            *********************************************************/

1127   1    declare
                  i             byte   initial (1),
                  last$dseg$byte byte  initial (0),
                  (vlow,vhigh)  byte;

            /*+++++++
```

```
1156  1          user$code = getuser;
1157  1          call set$dpb;                          /* get disk parameter blk */
1158  1          cdisk=cselect;                         /* get current disk */

1159  1          do while buff(i)=' ';
1160  2                 i = i + 1;
1161  2          end;
1162  1          buf$ptr = .buff(i);

1163  1          if buff(i) = '[' then do;              /* first, options */
1165  2                 buf$ptr = buf$ptr + 1;
1166  2                 call parse$options;            /* delimiter = ] or
                                                          null if end of cmd tail */

1167  2                 if delimiter = RBRACKET then call getfname;
1169  2                 else do;
1170  3                   call fill(.cmd(1),' ',26);     /* blank out command line */
1171  3                   cmd(0) = 0;
1172  3                 end;
1173  2          end;
1174  1          else do;                               /* filename ? */
1175  2                 call getfilename(.buff(1));    /* will set multi */

1176  2                 if optdel then do;
1178  3                        buf$ptr = ibp;
1179  3                        call parseoptions;
1180  3                 end;
1181  2                 else do;
1182  3                        call errprint(.errNOPT);
1183  3                        call terminate;
1184  3                 end;
1185  2                 call savewild;
1186  2          end;

1187  1          if option$map(opt$page) and option$map(opt$nopage) then do;
1189  2                 call errprint(.errPAGE);
1190  2                 call crlf;
1191  2                 PAGE = false;
1192  2          end;
1193  1          else if option$map(opt$nopage) then PAGE = false;
1195  1               else if option$map(opt$page) then PAGE = true;

                   if high(getscbword(COMbase)) = 0 then NONBANK = true;

1199  1          call exec;
1200  1          do while multi;
1201  2                 buf$ptr = ibp;
1202  2                 more = true;
1203  2                 call getfname;
1204  2                 call exec;
1205  2          end;

1206  1          call terminate;

1207  1     errver: call errprint(.errVERS);
1208  1          call terminate;
1209  1     end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

MODULE INFORMATION:

```
     CODE AREA SIZE     = 1E52H   7762D
     VARIABLE AREA SIZE = 01B1H    433D
     MAXIMUM STACK SIZE = 0016H     22D
     2202 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 SET
0000 MCDSOA
0004 BDISK    0080 BUFF      0080 BUFFA     0050 CMDRV     0000 CPU
007C CR             006D DOLLA     005C FCB       006C FCB16     005C FCBA
005C IFCB     005C IFCBA     0003 IOBYTE    0053 LEN0      0056 LEN1
0006 MAXB     0006 MEMSIZ    0005 MON1      0005 MON2      0005 MON2A
0005 MON3     0000 OFFSET    006E PARMA     0051 PASS0     0054 PASS1
007F RO             007D RR        007D RRECA     007F RRECO     005C SFCB
0080 TBUFF
0000 PARSE
02C9 DBLNK1   02BD DEBLNK    027B DELIM     02B6 DELIM1    02B9 DELIM2
0273 PADFLD   0211 PARSE1    0212 PARSE2    0180 PARSE     01A1 PARSE1
01CF PARSE2   01D2 PARSE3    01E4 PARSE4    01E8 PARSE5    01EA PARSE6
01FE PARSE7   0202 PARSE8    0223 PARSE9    024C PWFLD     025F PWFLD1
023E SETFD1   0240 SETFD2    0244 SETFD3    0229 SETFLD
0000 SET
2359 MEMORY   21A8 PAGE      0A4D PLM       02CD COPYRIGHT 02F2 NOTFOUND
0302 NOSPACE  0319 INVALID   032C DIRLABEL  033D OPTIONSET 034D READONLY
0357 RO             035D READWRITE 036D COMMA     0370 SETTO     0378 ERRORMSG
0380 READMODE 0385 WRITEMODE 038B DELETEMODE 0392 NOPASSWD  0397 ON
03A1 OFF      03AB LABELNAME 21A9 USERCODE  03B0 RESETMASK 0B79 PRINTCHAR
21AA CHAR     0B89 PRINTB    0B8F PRINTX    21AB A
0BB0 CHECKCONSTAT            0BB9 CRLF2     0BC4 TERMINATE 0BD0 CRLF
21AD CHARIN   0C1F PRINT     21AE A         0C31 GETVERSION 0C3A CONIN
0C43 SELECT   21B0 D         0C53 SEARCHFIRST             21B1 FCB
0C63 SEARCHNEXT       0C6C CSELECT    0C75 SETDMA    21B3 DMA       0C8$ WRITEPROT
0C8E GETUSER  0C97 RETURNERRORS              21B5 MODE      0CA7 SETIND
21B6 FCB      21B8 DPBA      0CBD SETDPB    0CC9 WRLBL     21BA FCB
0CDF GETLBL   21BC D         0CEF READXFCB  21BD FCB       0D05 WRXFCB
21BF FCB      0D1B RESETDRV  21C1 DRV       0D32 DELETE    21C2 FCB
21C4 FCBP     21C6 XFCB      21D2 XFCBMODE  21E6 CMD       2201 PASSWD
2212 SFACMD   2213 FILEREF   2214 LBLCMD    2215 XFCBCMD   2216 WILD
2217 OPTDEL   2218 MULTI     2219 NEWPASS   221A PASSMSG   221B NONBANK
221C PASSMODE 221D PASSWORD  221E MORE      221F IBP       2221 SAVDCNT
2223 SAVSEARCHA       2225 SAVSEARCHL 2226 DIRBUF    22A6 CDISK     22A7 VER
22A9 ERRORCODE 22AB PARSEFN  22AF LASTBUFFADR              03D0 ERRNOFILE
03F1 ERRDRIVEONLY            0409 ERRWASSPASS              042A ERRASSPASS
044A ERRFORMAT 049F ERRNOPROT 04C0 ERRUNREC  04D5 ERRNOMOD  04FD ERRUNRECM
0521 ERRVALM  0547 ERROPTMOD 0565 ERRBIGDEF 0597 ERRBIGNAME 05C4 ERRBIGPASS
05EE ERRCRAC  061E ERRSYSDIR 063B ERRRORW   0651 ERRNOPT    0667 ERRPAGE
069C ERRGLOBAL 06BF ERRDRVPROT 06EE ERRNBANK 0726 ERRVERS   0741 OPTMOD
07D9 OPTIONS  083A OFFOPT    084E MODS      086C OFFMODS   0873 ENDLIST
0874 ENDOFSTRING             0875 DELIMITERS 0890 SPACE     0891 RBRACKET
0892 ENDFF    0893 EQUAL     0894 LBRACKET  22B1 OPTIONMAP 22C4 MODSMAP
22D7 SFAMSG   22D8 DRVMSG    22D9 J         22DA STRINGPTR 22DC DEFPASS
22DE LABNAME  22E0 PASSNAME  22E2 LENDEF    22E3 LENPASS   22E4 LENLAB
22E5 BUFPTR   22E7 INDEX     22E8 ENDBUF    22E9 MINDEX    22EA DELIMITER
22EB SCBPD    22EF LINEPAGE  22F0 LINEOUT   22F1 SAVEFCB   2301 SAVEDCNT
2303 SAVESEARCHA             2305 SAVESEARCHL              2307 SAVEHASH1
2309 SAVEHASH2 230B SAVEHASH3 0D42 GETSCBWORD 230D OFF      0D5A SETSCB
230E OFF      230F VALUE     0D80 GETPAGE   0D91 PERROR    0DC6 PARSER
2311 P        0E20 FILL      2313 S         2315 F         2316 C
0E4B COPY     2317 S         2319 D         231B C         0E82 UCASE
231C C        0E9F ERRPRINT  231D MSG       0EB7 GETPASSWD 231F I
2320 C        0EBD RETRY     0ED6 NXTCHR    0F3F EXIT      0F46 PRINTDRV
0F55 PRINTFN  2321 K         0F8C BDOSERROR 2322 CODE      1008 SETSEARCH
2323 DCNT     101C SETUPFILE 2324 DIRINDEX  103F GETNEXT    2325 DCNT
2326 I        10AB PBOOL     2327 VALUE     10CC PRINTATT  115A ATTRIBUTE
2328 I        1174 RDATTRIBUTES            1193 SETDRVSTATUS
2329 FUNC     232A CODE      11EA READLABEL 232B MODE      232C DCNT
1261 READLBL  0895 D         1298 PUTFILE   12A5 SETUPXFCB
12FE DEFAULTPASS             133F SETPASSWORD              139D LNAME
232D I        1434 SETEXTENT 232E FUNCTION  232F MASKON    2330 MASKOFF
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
1487 PROTECT   2331 PMODE      1522 SETATT     2332 FUNC       2333 BYTES
1576 SHOWPASSWD        0896 LABEL1     1588 SHOWLBL    2334 MAKE       2335 ACCESS
1632 SHOWXFCB  16A5 PASSCHECK  2336 WHICH      1731 PUTATTRIBUTES
1799 WRITELABEL        17E4 WRITEXFCB  1828 WR0        1892 WR1        18AD SETDISK
18C7 WILDCARD  2337 I          18F1 SETUPFCB   2338 DCNT
195F CHECKCHOICE               233? INDEX      233A MINDEX     197B SEPARATOR
233B CHARACTER 233C K          1984 LOOP       19B3 OPTSCANNER 233D LISTPTR
233F OFFPTR    2341 I          2342 J          2343 WRDPOS     2344 CHARACTER
2345 LETTERINWORD              2346 FOUNDFIRST 2347 START      2348 INDEX
2349 SAVEINDEX 234A LENNEW     234B LENFOUND   234C VALID
1ACC CHECKINLIST               234D I          1B1F SETUP      1B3D TESTLETTER
1B7D SKIP      1BF4 EATBLANKS  1A0A EXIT1      1A1C GOOD       1A30 NEXTOPT
1A9F FINISHED  1C10 ERRORPRT   234E I          234F T
1C42 PARSEOPTIONS              2351 L          1C51 LOOP       1E22 ERROR1
1E2B ERROR2    1E34 ERROR3     1E3? ERROR4     1E46 ERROR5     1E4F MODPRT
1E58 OPTPRT    1E5E ERRPRT     1E6? DOOPTIONS  2352 DUMP       1F50 DO1
2008 SAVE      2039 SAVEWILD   2053 GETFILENAME                2353 BUFFADD
206F GETFNAME  2355 I          2356 LASTDSEGBYTE               2357 VLOW
2358 VHIGH     20C1 EXEC       0B6E ERRVER
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 SET#
0000 MCD80A#
0000 PARSE#
0000 SET#
0B79 22          0B7D 24          0B88 25          0B89 26          0B89 27
0B8E 28          0B8F 29          0B95 32          0B9E 33          0BA5 34
0BAC 35          0BAF 36          0BB0 37          0BB0 38          0BB9 39
0BB9 40          0BB9 41          0BBE 42          0BC3 43          0BC4 44
0BC4 45          0BC7 46          0BCF 47          0BD0 48          0BD0 50
0BD7 52          0BDB 53          0BE7 55          0BEA 56          0BED 57
0BF3 58          0BFA 59          0BFD 60          0C08 61          0C10 62
0C13 63          0C18 64          0C1B 65          0C1B 66          0C1B 67
0C1E 68          0C1F 69          0C25 71          0C28 72          0C30 73
0C31 74          0C31 75          0C3A 76          0C3A 77          0C3A 78
0C43 79          0C43 80          0C47 82          0C52 83          0C53 84
0C59 86          0C63 87          0C63 88          0C63 89          0C6C 90
0C6C 91          0C6C 92          0C75 93          0C75 94          0C7B 96
0C84 97          0C85 98          0C85 99          0C8E 100         0C8E 101
0C8E 102         0C97 103         0C97 104         0C9B 106         0CA6 107
0CA7 108         0CAD 110         0CB3 111         0CBD 112         0CBD 114
0CBD 115         0CC8 116         0CC9 117         0CCF 119         0CD5 120
0CDF 121         0CDF 122         0CE3 124         0CEF 125         0CEF 126
0CF5 128         0CFB 129         0D05 130         0D05 131         0D0B 133
0D11 134         0D1B 135         0D1B 136         0D1F 138         0D32 139
0D32 143         0D38 145         0D42 146         0D42 163         0D46 165
0D4C 166         0D51 167         0D5A 168         0D5A 169         0D62 171
0D68 172         0D6D 173         0D77 174         0D7F 175         0D80 176
0D80 177         0D85 178         0D88 179         0D91 180         0D91 181
0D91 182         0D97 183         0DA3 184         0DAE 185         0DB6 186
0DBC 187         0DC2 188         0DC5 189         0DC6 190         0DC6 193
0DCF 194         0DDC 195         0DE2 196         0DEE 198         0DF7 199
0DFF 200         0E08 201         0E0D 202         0E14 203         0E17 204
0E1C 205         0E20 206         0E20 207         0E2D 209         0E39 210
0E40 211         0E47 212         0E4A 213         0E4B 214         0E5A 217
0E66 218         0E70 219         0E77 220         0E7E 221         0EB1 222
0E82 223         0E82 225         0E8D 226         0E95 227         0E9B 228
0E9F 229         0E9F 230         0EA5 232         0EAB 233         0EB3 234
0EB6 235         0EB7 236         0EB7 238         0EBD 239         0EC8 240
0ED6 241         0EE1 242         0EF1 243         0EF9 244         0EFC 245
0F04 246         0F07 247         0F0F 249         0F17 250         0F1A 252
0F2A 253         0F2D 254         0F2D 255         0F2D 256         0F35 257
0F38 258         0F3F 259         0F45 260         0F46 261         0F46 262
0F4F 263         0F54 264         0F55 265         0F55 267         0F5B 268
0F66 269         0F6E 270         0F73 271         0F84 272         0F8B 273
0F8C 274         0F8C 276         0F92 277         0F9E 279         0FA4 280
0FA7 281         0FAA 282         0FB2 283         0FB8 284         0FC0 286
0FC6 287         0FCC 288         0FCC 289         0FCF 290         0FCF 291
0FD7 292         0FDD 293         0FE5 294         0FEB 295         0FF3 296
0FF9 297         1001 298         1007 299         1008 300         100C 302
1012 303         101B 304         101C 305         1020 307         1028 309
103A 310         103E 311         103E 312         103F 313         103F 315
1049 316         104F 317         1058 318         1061 319         106A 320
1073 321         107C 322         1085 323         1093 324         109E 325
10A1 326         10A8 327         10AB 328         10AB 329         10AF 331
10B5 332         10BC 333         10C5 334         10CB 335         10CC 336
115A 337         115E 339         116E 340         1171 341         1174 342
10CC 343         10D2 344         10DB 345         10E4 346         10EA 347
10F0 348         10F9 350         10FF 351         1105 352         1108 353
110E 354         1113 355         111C 356         1121 357         112A 358
112F 359         1138 360         113D 361         1146 362         114B 363
1154 364         1159 365         1174 366         1174 367         117B 368
1182 370         1188 371         1192 372         1192 373         1193 374
1197 376         119F 377         11A8 378         11B2 379         11BA 381
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

11C0 382 11C3 383 11C6 384 11CC 385 11D4 387
11DA 388 11E0 389 11E3 390 11E9 391 11E9 392
11EA 393 1261 395 1261 397 1267 398 1270 399
1278 400 128D 401 128E 402 1294 403 1297 404
11EA 405 11F1 406 11F2 407 11FC 408 1201 409
120A 411 120D 412 121F 413 1227 414 1234 415
1241 416 1246 417 1249 419 124F 420 125B 421
125B 422 1260 423 1298 424 1298 425 129B 426
129E 427 12A1 428 12A4 429 12A5 430 12A5 431
12AC 433 12B1 434 12BD 435 12C5 436 12D4 438
12DD 439 12E3 441 12E9 442 12EC 443 12EC 444
12ED 445 12ED 446 12F3 447 12F8 448 12FD 449
12FD 450 12FE 451 12FE 452 1305 454 130B 455
130C 456 130C 457 1317 458 1325 459 132D 460
1333 461 1338 462 133E 463 133F 464 133F 465
1346 467 134D 469 1353 470 1354 471 1354 472
1357 473 135F 474 1362 476 1365 477 1370 478
1370 479 137B 480 1383 482 1388 483 1389 484
1389 485 138E 486 139C 487 139D 488 139D 490
13A4 491 13A5 492 13AC 494 13B2 495 13B7 496
13B8 497 13B8 498 13BB 499 13C7 500 13D0 502
13E0 503 13F0 505 1401 506 141A 507 141B 508
141B 509 1422 510 1433 511 1433 512 1434 513
143F 515 1446 516 1447 517 144E 519 1453 520
1459 521 145A 522 145A 523 145D 524 146B 525
147A 526 1486 527 1487 528 1487 529 1487 531
148E 533 1491 534 1497 535 149C 536 14A4 537
14AC 538 14B4 539 14BC 540 14C4 542 14C9 543
14D4 544 14D4 545 14DB 546 14E3 547 14E6 549
14ED 551 14F3 552 14F4 553 14F4 554 14FA 555
1501 557 1507 558 1508 559 1508 560 1512 561
1521 562 1521 563 1522 564 1528 566 152F 567
1530 568 1537 570 153C 571 1542 572 1543 573
1543 574 1551 575 1562 576 1570 577 1575 578
1576 579 1576 580 157C 581 1581 582 1587 583
1588 585 1588 587 158E 588 1591 589 1594 590
159A 591 159D 592 15AC 593 15B5 594 15BB 595
15CD 596 15D9 597 15E2 598 15EB 599 15EF 600
15F8 601 15FE 602 160D 603 1616 604 161C 605
161F 606 162B 608 162E 609 1631 610 1631 611
1632 612 1632 613 1638 614 1642 615 164B 616
1655 617 165E 618 1668 619 1671 620 1685 621
168E 622 1694 623 169B 625 16A1 626 16A4 627
16A4 628 16A5 629 16A9 631 16B2 633 16B5 634
16BD 635 16C3 636 16C9 637 16CC 638 16D3 639
16D6 640 16DE 641 16EC 642 16F4 643 1702 644
170B 645 1714 647 1717 648 171F 649 1728 650
172B 651 172E 652 172E 653 172E 654 1731 655
1731 656 1731 657 173C 658 1745 659 174E 661
1757 662 1758 663 1761 665 1764 666 1767 667
1768 668 1768 669 176B 671 1771 672 1774 673
1774 674 177D 675 1792 677 1795 678 1798 679
1798 680 1799 681 1799 682 17A4 683 17AD 684
17B6 686 17BF 687 17C0 688 17C9 690 17CC 691
17D2 692 17D3 693 17D3 694 17D6 695 17D9 697
17DF 698 17E0 699 17E0 700 17E3 701 17E4 702
17E4 703 17ED 705 17F4 706 17F7 707 17FE 708
1801 709 1808 710 1809 711 1810 712 1819 714
181F 715 1822 716 1822 717 1827 718 1828 719
1828 720 1830 721 1837 722 183F 723 1845 724
184E 725 1857 726 1860 728 1869 729 186A 730
1873 732 1876 733 1879 734 187A 735 187A 736
187D 738 1883 739 1889 740 188A 741 188A 742
1892 744 189A 745 18A6 746 18A6 747 18A9 748
18AC 749 18AD 750 18AD 751 18B5 753 18BC 754
18C3 755 18C6 756 18C6 757 18C7 758 18C7 760

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

18D5 761 18E4 762 18E7 763 18EE 764 18F1 765
18F1 766 18F1 768 18F? 769 1900 770 190C 771
1924 773 1929 774 192F 775 193D 777 1943 778
1949 779 194C 780 194F 781 194F 782 1956 783
1959 784 195E 785 195F 786 1965 788 197B 789
197B 790 197F 792 1?8A 793 1994 794 1997 795
19A7 796 19AB 797 19AF 798 19B2 799 19B3 800
1ACC 803 1ACC 805 1AD2 806 1ADD 807 1AEE 808
1AF2 809 1AF6 810 1B07 811 1B08 812 1B16 813
1B19 814 1B1E 815 1B1F 816 1B1F 817 1B1F 818
1B26 819 1B29 820 1B2F 821 1B34 822 1B3C 823
1B3D 824 1B3D 825 1B41 826 1B4F 827 1B57 828
1B68 829 1B6D 830 1B71 831 1B7C 832 1B7D 833
1B7D 834 1B8B 835 1B95 836 1BC5 837 1BC9 838
1BD4 839 1BDE 840 1BE1 841 1BE7 842 1BF3 843
1BF4 844 1BF4 845 1BF4 847 1C05 848 1C0C 849
1C0F 850 19BD 851 19C0 852 19C5 853 19C8 854
19D0 855 19D6 856 19E0 857 19E5 858 19ED 859
19F0 860 19F7 861 19FA 862 1A04 863 1A07 864
1A0A 865 1A0D 866 1A10 867 1A16 868 1A19 869
1A1C 870 1A22 871 1A25 872 1A2B 873 1A30 874
1A36 875 1A39 876 1A41 877 1A44 878 1A64 879
1A6B 881 1A70 882 1A7F 883 1A82 884 1A89 885
1A8C 886 1A93 887 1A96 888 1A99 889 1A9C 890
1A9F 891 1AAD 892 1AB0 893 1AB8 894 1AC2 895
1AC8 896 1ACC 897 1C10 898 1C10 900 1C1D 901
1C2C 902 1C33 903 1C3A 904 1C41 905 1C42 906
1C42 908 1C47 909 1C4C 910 1C51 911 1C59 912
1C5A 913 1C64 914 1C65 915 1C6F 916 1C70 917
1C81 918 1C84 919 1C9E 921 1CA8 923 1CBB 924
1CBE 925 1CD5 927 1CE6 928 1CE9 929 1D04 930
1D07 931 1D0A 933 1D10 934 1D15 935 1D18 936
1D1D 937 1D25 938 1D2F 939 1D36 940 1D3A 941
1D3D 942 1D47 944 1D51 945 1D58 946 1D58 947
1D5C 948 1D62 950 1D6B 952 1D71 953 1D7E 955
1D84 956 1D89 957 1D89 958 1D8C 959 1D95 961
1D9B 962 1DA8 964 1DAD 965 1DB3 966 1DB3 967
1DB6 969 1DBC 970 1DC9 972 1DCF 973 1DD4 974
1DD4 975 1DD4 976 1DD4 977 1DD4 978 1DD4 979
1DD4 980 1DF3 981 1DF6 982 1E03 983 1E0C 984
1E1F 985 1E22 986 1E28 987 1E2B 988 1E31 989
1E34 990 1E3A 991 1E3D 992 1E43 993 1E46 994
1E4C 995 1E4F 996 1E55 997 1E58 998 1E5E 999
1E61 1000 1E64 1001 1E65 1002 1E65 1004 1E6C 1005
1E73 1006 1E7A 1007 1E81 1008 1E88 1009 1E8F 1010
1E96 1011 1E9D 1012 1EA4 1013 1EAB 1014 1EB2 1015
1EB5 1016 1EBC 1017 1EBF 1018 1EC6 1019 1EC9 1020
1ED0 1021 1ED3 1022 1EDE 1024 1EE9 1026 1EF0 1027
1EF6 1028 1EFC 1029 1EFF 1030 1F02 1031 1F02 1032
1F02 1033 1F09 1035 1F10 1037 1F15 1038 1F1F 1039
1F1F 1040 1F29 1041 1F29 1042 1F30 1044 1F37 1046
1F3C 1047 1F46 1048 1F46 1049 1F50 1050 1F50 1051
1F5B 1053 1F62 1054 1F68 1055 1F6E 1056 1F71 1057
1F74 1059 1F7B 1060 1F85 1061 1F8C 1063 1F91 1064
1F98 1065 1F98 1066 1F98 1067 1F9F 1068 1FA9 1069
1FB4 1071 1FBA 1072 1FBD 1073 1FC0 1075 1FC7 1076
1FCE 1078 1FD3 1079 1FDA 1080 1FDD 1081 1FE5 1082
1FEC 1083 1FF3 1085 1FF8 1086 1FFF 1087 2002 1088
2007 1089 2007 1090 2008 1091 2008 1092 2010 1093
2018 1094 2020 1095 2028 1096 2030 1097 2038 1098
2039 1099 2039 1100 2043 1101 204F 1102 2052 1103
2053 1104 2059 1106 205F 1107 2062 1108 2068 1109
206E 1110 206F 1111 206F 1112 2077 1113 207E 1115
2084 1116 2089 1117 208F 1118 2095 1119 2098 1120
2098 1121 20A4 1122 20AC 1123 20B1 1124 20BD 1125
20C0 1126 20C1 1128 20C1 1129 20C8 1130 20CE 1131

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
20D2 1132   20D5 1133   20DA 1134   20E1 1135   20E7 1137
20EE 1138   20F1 1139   20F8 1140   20FB 1141   20FB 1142
2100 1143   2107 1144   210F 1145   2116 1146   211B 1147
211E 1148   0A4A 1149   0A5C 1150   0A5D 1151   0A61 1152
0A67 1153   0A6C 1154   0A7A 1155   0A77 1156   0A7D 1157
0A80 1158   0A86 1159   0A95 1160   0A99 1161   0A9C 1162
0AA8 1163   0AB4 1165   0ABB 1166   0ABE 1167   0AC8 1168
0ACE 1170   0AD9 1171   0ADE 1172   0ADE 1173   0AE1 1175
0AE7 1176   0AEE 1178   0AF4 1179   0AF7 1180   0AFA 1182
0B00 1183   0B03 1184   0B03 1185   0B06 1186   0B06 1187
0B11 1189   0B17 1190   0B1A 1191   0B1F 1192   0B22 1193
0B29 1194   0B31 1195   0B38 1196   0B3D 1197   0B48 1198
0B4D 1199   0B50 1200   0B57 1201   0B5D 1202   0B62 1203
0B65 1204   0B68 1205   0B6B 1206   0B6E 1207   0B74 1208
0B77 1209
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                ;Function 100 RSX (set/create directory label
                ;       Only for Non banked systems
                ;
                ;       Procedure:
                ;               1. If this BDOS call <> f100 then go to NEXT
                ;               2. select the current disk for BIOS calls
                ;               3. search for current label
                ;               4. if no label then do
                ;                  a. find first empty dir slot
                ;                  b. if no empties then return error
                ;                  c. create dir label from user FCB in DE
                ;                  d. call update SFCB
                ;                  e. return
                ;               5. if password protected then ok = password()
                ;               6. if <>ok then return error
                ;               7. update label from user info
                ;               8. call update SFCB
                ;               9. return
                ;
                ;                                               P. Balma
                ;
                ;                       RSX     PREFIX
                ;
0000 0000000000serial:  db      0,0,0,0,0,0
0006 C31B00     jmp1:   jmp     ftest
0009 C3         NEXTj:  db      0c3h            ; next RSX or BDOS
000A 0000       NEXTa:  db      0,0             ; next address
000C 0000       prev:   dw      0               ; where from
000E FF         remove: db      0ffh            ; remove RSX at warm start
000F FF         nbank:  db      0FFh            ; non banked RSX
0010 4449524C42rsxname: db      'DIRLBL '
0018 0000       space:  dw      0
001A 00         patch:  db      0
                ;
                ;
                ftest:
001B F5                 push a                  ;save user regs
001C 79                 mov a,c
001D FE64               cpi 64h                 ;compare BDOS func 100
001F CA2700             jz func100
0022 F1                 pop a                   ;some other BDOS call
                goto$next:
0023 2A0A00             lhld NEXTa              ; go to next and don't return
0026 E9                 pchl

                        ; Set directory label
                        ; de -> .fcb
                        ;       drive location
                        ;       name & type fields user's discretion
                        ;       ex'nt field definition
                        ;               bit 1 (80h): enable passwords on drive
                        ;               bit 2 (40h): enable file access
                        ;               bit 3 (20h): enable file update stamping
                        ;               bit 4 (10h): enable file create stamping
                        ;               bit 8 (01h): assign new password to dir lbl
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                func100:
0027 F1             pop a
0028 2100003922     lxi h,0 ! dad sp ! shld ret$stack    ; save user stack
002F 311E04         lxi sp,loc$stack
                                                        COPYRIGHT © 1982
                                                        DIGITAL RESEARCH
                                                        P. O. BOX 579
                                                        PACIFIC GROVE, CA 93950
                                                        SER. #
0032 EB22A503EB     xchg ! shld info ! xchg
0037 0E19CD2300     mvi c,19h ! call goto$next ! sta curdsk ; get current disk
003F 0E1DCD2300     mvi c,1dh ! call goto$next          ; is drive R/O ?
0044 3A92034FCD     lda curdsk ! mov c,a ! call hlrotr
004B 7DE601C22A     mov a,l ! ani 01h ! jnz read$only

0051 2AA503CD49     lhld info ! call getexta ! push a   ; if user tries to set time
0058 E670329303     ani 0111$0000b ! sta set$time       ; stamps and no SFCB's,..error
005D 7EE67F77       mov a,m ! ani 7fh ! mov m,a         ; mask off password bit
0061 E601329503     ani 1 ! sta newpass                 ; but label can have password

0066 0E69D51199     mvi c,69h ! push d ! lxi d,stamp    ; get time for possible
006C CD2300D1       call goto$next ! pop d              ; update later

0070 0E3111BD03     mvi c,31h ! lxi d,SCBPB ! call goto$next; get BDOS current dma
0078 22B303         shld curdma

007B 3A9203CD52     lda curdsk ! call dsksel            ; BIOS select and sets
                                                        ; disk parameters
                                                        ; Does dir lbl exist on drive?
0081 CDB601         call search                         ; return if found or
0084 E50600         push h ! mvi b,0                    ; successfully made
0087 1120003A9E     lxi d,20h ! lda nfcbs ! mov c,a     ; Are there SFCB's in directory
008E 197EFE21CA main0: dad d ! mov a,m ! cpi 21h ! jz main1
0095 043A9F033C     inr b ! lda i ! inr a ! sta i ! cmp c
009E C28E00         jnz main0

00A1 3A9303B7C2     lda set$time ! ora a ! jnz no$SFCB  ; no, but user wants to set
                                                        ; time stamp
00A8 32A103         sta SFCB                            ; SFCB = false

00AB 22A3037832 main1: shld SFCB$addr ! mov a,b ! sta j ! lhld info
00B5 EBE1E523       xchg ! pop h ! push h ! inx h       ; HL => dir FCB, DE => user FCB
00B9 130E0C         inx d ! mvi c,0ch                   ; prepare to move DE to HL
00BC CD4F033A95     call move ! lda newpass             ; find out if new password ?
00C2 B7             ora a
00C3 C4BC02         cnz scramble                        ; scramble user pass & put in
                                                        ; dFCB

00C6 3AA1033CC2     lda SFCB ! inr a ! jnz mainx1       ; any SFCB's


                main2:                                  ; update time & date stamp
00CD 3AA003473E     lda j ! mov b,a ! mvi a,2           ; j = FCB position from  SFCB
00D3 90             sub b                               ; in 4 FCB sector (0,1,2), thus
                                                        ;  FCBx - 2
                                                        ;  FCBy - 1
                                                        ;  FCBz - 0
                                                        ;  SFCB
```

```
                                                    ; So, 2-j gives FCB offset in
                                                    ; SFCB

00D4 06004F2AA3         mvi b,0 ! mov c,a ! lhld SFCB$addr
00DA 23110A000C         inx h ! lxi d,0ah ! inr c
00DF 0DCAE700  mainx0:  dcr c ! jz mainx1
00E3 19C3DF00           dad d ! jmp mainx0

00E7 D1D5E5    mainx1:  pop d ! push d ! push h            ; HL => dFCB
00EA EB11180019         xchg ! lxi d,18h ! dad d           ; HL => dfcb(24) (TS field)
00EF EBE1D5             xchg ! pop h ! push d              ; of DIR LABEL
                                                           ; HL => Time/stamp pos in SFCB
00F2 3A9D033CC2         lda NEW ! inr a ! jnz st0          ; did we create a new DL?
00F9 CD7403C30D         call stamper ! jmp st1             ; yes

00FF 11040019  st0:     lxi d,4 ! dad d                    ; update time stamp
0103 D1E5EB1104         pop d ! push h ! xchg ! lxi d,4    ; DFCB position
0109 19EBE1D5           dad d ! xchg ! pop h ! push d
010D CD7403    st1:     call stamper
0110 E1                 pop h

0111 E1CD4903F6mainr:   pop h ! call getexta ! ori 1 ! mov m,a  ; set lsb extent
0118 CDFD02             call write$dir
011B AF210000C3         xra a ! lxi h,0 !jmp goback        ; no SFCB, so finished

                no$SFCB:
0122 3EFF21FF00         mvi a,0ffh ! lxi h,0ffh ! jmp goback

                read$only:
012A 3EFF21FF02         mvi a,0ffh ! lxi h,02ffh

012F E52AB50336goback:  push h ! lhld aDIRBCB ! mvi m,0ffh ; tell BDOS not to use buffer
                                                           ; contents
0135 F5                 push a

0136 0E0DCD2300         mvi c,0dh ! call goto$next         ; BDOS reset
013B 3A92035F0E         lda curdsk ! mov e,a ! mvi c,0eh
0141 CD2300             call goto$next
0144 3A9203CDC1         lda curdsk ! call seldsk           ; restore BDOS environment
014A F1D1               pop a ! pop d
014C 2AFC03F9           lhld ret$stack ! sphl              ; restore user stack
0150 EB                 xchg                               ; move error return to h
0151 C9                 ret


                dsksel:                             ; select disk and get parameters

0152 CDC103             call seldsk                        ; Bios select disk
0155 CD5903             call gethl                         ; DE = XLT addr
0158 22AB03EB           shld XLT ! xchg
015C 010A0009           lxi b,0ah ! dad b                  ; HL = addr DPB
0160 CD5903             call gethl
0163 22B703EB           shld aDPB ! xchg
0167 01040009           lxi b,4 ! dad b                    ; HL = addr DIR BCB
016B CD590322B5         call gethl ! shld aDIRBCB
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
169  0171 010A0009          lxi b,0ah ! dad b                    ; Hl => DIR buffer
170  0175 22AD03            shld bufptr                          ; use BDOS buffer for
171                                                              ; BIOS reads & writes
172                                                              ; must jam FF into it to
173                                                              ; signal don't use when done
174  0178 2AB703            lhld aDPB
175  017B CD5903            call gethl                           ; get [HL]
176  017E 22AF03EB          shld spt ! xchg
177  0182 2323232323        inx h! inx h! inx h ! inx h! inx h!   ; HL => dirmax
178  0187 CD590322BB        call gethl ! shld dirmax ! xchg
179  018E 2323              inx h ! inx h !
180  0190 CD590322A7        call gethl ! shld checkv ! xchg
181  0197 CD590322A9        call gethl ! shld offset ! xchg
182                                                              ; HL => phys shift
183  019E CD5903EB          call gethl ! xchg                    ; E = physhf, D = phymsk
184  01A2 147A              inr d ! mov a,d                      ; phys mask+1 = # 128 byte rcd
185                                                              ; phymsk * 4 = nfcbs/rcd
186  01A4 B717B717          ora a ! ral ! ora a ! ral            ; clear carry & shift phymsk
187  01A8 329E03            sta nfcbs
188
189  01AB 2AAF03            lhld spt                             ; spt = spt/phymsk
190  01AE 4BCD6703          mov c,e ! call hlrotr                ; => spt = shl(spt,physhf)
191  01B2 22AF03            shld spt
192  01B5 C9                ret
193
194                search:                                       ; search dir for pattern in
195                                                              ; info of length in
196  01B6 AF32970332        xra a ! sta sect ! sta empty
197  01BD 21000022B1        lxi h,0 ! shld dcnt
198
199  01C3 2AAD03444D        lhld bufptr ! mov b,h ! mov c,l      ; set BIOS dma
200  01C8 CDEE03            call setdma
201
202  01CB CDF302      src0: call read$dir
203  01CE FE00C23602        cpi 0 ! jnz oops                     ; if A <> 0 then BIOS error
204
205  01D3 06003A9E03        mvi b,0 ! lda nfcbs ! mov c,a        ; BC always = nfcbs
206
207  01D9 2AAD031120        lhld bufptr ! lxi d,20h              ; start of buffer and FCB
208  01DF AF                xra a                                ; do i = 0 to nfcbs - 1
209  01E0 329F037E    src1: sta i ! mov a,m                      ; user #
210  01E4 FE20C2FA01        cpi 20h ! jnz src2                   ; dir label mark
211
212  01E9 E511100019        push h ! lxi d,10h ! dad d ! mov a,m  ; found label, move to DM to
213  01EF B7E1C8            ora a ! pop h ! rz                   ; check if label is pass prot
214  01F2 E5FE20E1C2        push h ! cpi 20h ! pop h ! jnz checkpass
215  01F9 C9                ret
216
217  01FA 3A98033CCA  src2: lda empty ! inr a ! jz src3          ; record first sect with empty
218  0201 7E                mov a,m
219  0202 FEE5C21202        cpi 0e5h ! jnz src3 ! lda sect       ; save sector #
220  020A 32A2033EFF        sta savsect ! mvi a,0ffh ! sta empty ; set empty found = true
221  0212 19          src3: dad d                                ; position to next FCB
222  0213 3A9F033C          lda i ! inr a                        ; while i < nfcbs
223  0217 B9C2E001          cmp c ! jnz src1
224
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
021B 2ABB03EB2A       lhld dirmax ! xchg ! lhld dcnt          ; while (dcnt < dirmax) &
                                                              ; dir label not found
0222 0922B103CD       dad b ! shld dcnt ! call subdh          ; is dcnt <= dirmax ?
0229 DA3F02           jc not$found                            ; no
022C 3A97033C32       lda sect ! inr a ! sta sect ! jmp src0

0236 3EFF21FF01oops:  mvi a,0ffh ! lxi h,1ffh
023B C1C32F01         pop b ! jmp goback                      ; return perm. error

                not$found:                                    ; must make a label

023F 3A98033CC2       lda empty ! inr a ! jnz no$space        ; if empty = false...
0246 3AA2033297       lda savsect ! sta sect
024C CDF302           call read$dir                           ; get sector
024F 2AAD031120       lhld bufptr ! lxi d,20h ! mvi c,0       ; C = FCB offset in buffer
0257 7EFEE5CA62 nf0:  mov a,m ! cpi 0e5h ! jz nf1
025D 190CC35702       dad d ! inr c !jmp nf0                  ; Know that empty occurs here
                                                              ; so don't need bounds test
0262 362079329F nf1:  mvi m,20h ! mov a,c ! sta i
0268 3E00E50E20       mvi a,0 ! push h ! mvi c,32             ; clear fcb to spaces
026D 230DCA7602 nf2:  inx h ! dcr c ! jz nf3
0272 77C36D02         mov m,a ! jmp nf2
0276 E1         nf3:  pop h
0277 3EFF329D03       mvi a,0ffh ! sta NEW
027C C9               ret                                     ; HL => dir FCB

027D 3EFF21FF00no$space: mvi a,0ffh ! lxi h,0ffh ! pop b ! jmp goback

                check$pass:                   ; Dir is password protected, check dma for
                                              ; proper password

0286 E5               push h                          ; save addr dir FCB
0287 110D00194E       lxi d,0dh ! dad d ! mov c,m     ; get XOR sum in S1, C = S1
028C 110A0019         lxi d,0ah ! dad d               ; position to last char in label pass
0290 0608             mvi b,8                         ; # chars in pass
0292 EB2AB303EB       xchg ! lhld curdma ! xchg       ; DE => user pass, HL => label pass

0297 7EA9C5     cp0:  mov a,m ! xra c ! push b        ; HL = XOR(HL,C)
029A 4F1AB9           mov c,a ! ldax d ! cmp c        ; compare user and label passwords
029D C2B202           jnz wrong$pass
02A0 C1132B05         pop b ! inx d ! dcx h ! dcr b
02A4 C29702           jnz cp0

02A7 EB22B303         xchg ! shld curdma              ; curdma = 2nd pass in field if there
02AB E1               pop h                           ; restore dir FCB addr
02AC 3EFF329403       mvi a,0ffh ! sta oldpass
02B1 C9               ret

                wrong$pass:
02B2 3EFF21FF07       mvi a,0ffh ! lxi h,07ffh ! pop b ! pop b
02B9 C32F01           jmp goback

                scramble:                             ; encrypt password at curdma
                                                      ; 1. sum each char of pass.
                                                      ; 2. XOR each char with sum
                                                      ; 3. reverse order of encrypted pass
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
                                ;checkpass sets to 2nd pos if
02BC 0108002AB3         lxi b,8 ! lhld curdma
02C2 3A94033CCA         lda oldpass ! inr a ! jz scr0           ;old pass else must move dma
02C9 0922B303           dad b ! shld curdma
                                                                ; B = sum, C = max size of pass
02CD 7E80470D   scr0:   mov a,m ! add b ! mov b,a ! dcr c
02D1 23C2CD02           inx h ! jnz scr0


02D5 D1E1D5             pop d ! pop h ! push d                  ; H => dFCB, D was return
02D8 110D001970         lxi d,0dh ! dad d ! mov m,b             ; S1 = sum
02DD 110A0019           lxi d,0ah ! dad d                       ; position to last char in pass
02E1 0E08EB2AB3         mvi c,8 ! xchg ! lhld curdma
02E7 7EA8EB77   scr1:   mov a,m ! xra b ! xchg ! mov m,a        ; XOR(char) => dFCB
02EB EB231B0DC2         xchg ! inx h ! dcx d ! dcr c ! jnz scr1

02F2 C9                 ret


                read$dir:                               ; read directory into bufptr

02F3 CD0A03             call track
02F6 CD1903             call sector
02F9 CDD803             call rdsec
02FC C9                 ret

                writedir:                               ; write directory from bufptr
02FD 3A9703             lda sect
0300 CD0A03             call track
0303 CD1903             call sector
0306 CDDE03             call wrsec
0309 C9                 ret

                track:                                  ; set the track for the BIOS call

030A 2AAF03CD38         lhld spt ! call intdiv                  ; E = integer(sect/spt)
0310 2AA90319EB         lhld offset ! dad d ! xchg ! call settrk
0318 C9                 ret

                sector:                                 ; set the sector for the BIOS
0319 3A9703             lda sect
031C 2AAF03CD38         lhld spt ! call intdiv          ; get mod(sect,spt)
0322 7995               mov a,c ! sub l                 ; D = x * spt such that D > sect
                                                        ; D - spt = least x*spt s.t. D < sect
0324 4F3A970391         mov c,a ! lda sect ! sub c      ; a => remainder of sect/spt
0329 06004F2AAB         mvi b,0 ! mov c,a ! lhld XLT    ; BC = logical sector #, DE = translate
032F EBCDE403           xchg ! call sectrn              ; table address
0333 EBCDD003           xchg ! call setsec              ; BC = physical sector #
0337 C9                 ret


                intdiv:                                 ; compute the integer division of A/L

033B 0E00110000         mvi c,0 ! lxi d,0
033D F5         int0:   push a                          ; compute the additive sum of L such
033E 7D814F             mov a,l ! add c ! mov c,a       ; that C = E*L where C = 1,2,3,...
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
0341 F1              pop a
                     
0342 B91CD23D03      cmp c ! inr e ! jnc int0        ; if A < E%L then return E - 1
0347 1D              dcr e
0348 C9              ret
                     
             getexta:
                                  ; Get current extent field address to hl
0349 110C0019        lxi d,0ch ! dad d               ; hl=.fcb(extnum)
034D 7E              mov a,m
034E C9              ret
                     
             move:                ; Move data length of length c from source de to
                                  ; destination given by hl
                     
034F 0C              inr c                                  ; in case it is zero
                     move0:
0350 0DC8                dcr c! rz                          ; more to move
0352 1A77                ldax d! mov m,a                    ; one byte moved
0354 1323                inx d! inx h                       ; to next byte
0356 C35003              jmp move0
                     
             gethl:                                  ; get the word pointed at by HL
0359 5E235623        mov e,m ! inx h ! mov d,m ! inx h
035D EBC9            xchg ! ret
                     
             subdh:                          ; HL = DE - HL
                     
035F B7              ora a                              ;clear carry
0360 7B956F          mov a,e ! sub l ! mov l,a
0363 7A9C67          mov a,d ! sbb h ! mov h,a
0366 C9              ret
                     
             hlrotr:
                                                ; rotate HL right by amount c
0367 0C              inr c                              ; in case zero
0368 0DC8       hlr: dcr c! rz                          ; return when zero
036A 7CB71F67        mov a,h! ora a! rar! mov h,a       ; high byte
036E 7D1F6F          mov a,l! rar! mov l,a              ; low byte
0371 C36803          jmp hlr
                     
             stamper:                                ; move time stamp into SFCB & FCB
0374 3AA1033C        lda SFCB ! inr a                   ; no SFCB, update DL only
0378 CC8903C1D1      cz stmp ! pop b ! pop d ! push h ! xchg
037F C5CD8903C1      push b ! call stmp ! pop b ! xchg ! pop h ! push d
0387 C5              push b
0388 C9              ret
0389 1199030E04stmp: lxi d,stamp ! mvi c,4 ! call move
0391 C9              ret
                     
             ;*****************************************************************
                     
0392 00      curdsk:     db      0
0393 00      set$time:   db      0
0394 00      oldpass:    db      0
0395 00      newpass:    db      0
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
0396 00       pass$prot       db      0
0397 00       sect:           db      0
0398 00       ...ty:          db      0
0399          stamp:          ds      4
039D 00       NEW:            db      0
039E 00       nfcbs:          db      0
039F 00       i:              db      0
03A0 00       j:              db      0
03A1 FF       SFCB:           db      0ffh
03A2 00       savsect:        db      0

03A3 0000     SFCB$addr:      dw      0
03A5 0000     ...fo:          dw      0
03A7 0000     checkv          dw      0
03A9 0000     offset:         dw      0
03AB 0000     XLT:            dw      0
03AD 0000     bufptr:         dw      0
03AF 0000     spt:            dw      0
03B1 0000     d...t:          dw      0
03B3 0000     curdma:         dw      0
03B5 0000     aDIRBCB         dw      0
03B7 0000     aDPB:           dw      0
03B9 0000     dFCB:           dw      0
03BB 0000     dirmax:         dw      0

              SCBPB:
03BD 3C       Soff:           db      3ch
03BE 00       Sset:           db      0
03BF 0000     Svalue:         dw      0

              ;
              ;***************************************************************
              ;*                                                             *
              ;*      bios calls from for track, sector io                   *
              ;*                                                             *
              ;***************************************************************
              ;***************************************************************
              ;*                                                             *
              ;*        equates for interface to cp/m bios                   *
              ;*                                                             *
              ;***************************************************************
              ;
              ;
0000 =        base    equ     0
0001 =        wboot   equ     base+1h ;warm boot entry point stored here
0018 =        sdsk    equ     18h     ;bios select disk entry point
001B =        strk    equ     1bh     ;bios set track entry point
001E =        ssec    equ     1eh     ;bios set sector entry point
0021 =        stdma   equ     21h
0024 =        read    equ     24h     ;bios read sector entry point
0027 =        write   equ     27h     ;bios write sector entry point
002D =        stran   equ     2dh     ;bios sector translation entry point
              ;
              ;***************************************************************
              ;*                                                             *
              ;***************************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
449                     seldsk: ;select drive number 0-15, in C
450                             ;1-> drive no.
451                             ;returns-> pointer to translate table in HL
452   03C1 4F                   mov c,a         ;c = drive no.
453   03C2 111800               lxi d,sdsk
454   03C5 C3F403               jmp gobios
455                     ;
456                     ;*************************************************************
457                     ;*                                                           *
458                     ;*************************************************************
459                     settrk: ;set track number 0-76, 0-65535 in BC
460                             ;1-> track no.
461   03C8 42                   mov b,d
462   03C9 4B                   mov c,e         ;bc = track no.
463   03CA 111B00               lxi d,strk
464   03CD C3F403               jmp gobios
465                     ;
466                     ;*************************************************************
467                     ;*                                                           *
468                     ;*************************************************************
469                     setsec: ;set sector number 1 - sectors per track
470                             ;1-> sector no.
471   03D0 42                   mov b,d
472   03D1 4B                   mov c,e         ;bc = sector no.
473   03D2 111E00               lxi d,ssec
474   03D5 C3F403               jmp gobios
475                     ;
476                     ;*************************************************************
477                     ;*                                                           *
478                     ;*************************************************************
479                     rdsec:  ;read current sector into sector at dma addr
480                             ;returns in A register: 0 if no errors
481                             ;                       1 non-recoverable error
482   03D8 112400               lxi d,read
483   03DB C3F403               jmp gobios
484                     ;*************************************************************
485                     ;*                                                           *
486                     ;*************************************************************
487                     wrsec:  ;writes contents of sector at dma addr to current sector
488                             ;returns in A register: 0 errors occured
489                             ;                       1 non-recoverable error
490   03DE 112700               lxi d,write
491   03E1 C3F403               jmp gobios
492                     ;
493                     ;*************************************************************
494                     ;*                                                           *
495                     ;*************************************************************
496                     sectrn: ;translate sector number
497                             ;1-> logical sector number (fixed(15))
498                             ;2-> pointer to translate table
499                             ;returns-> physical sector number
500   03E4 D5                   push d
501   03E5 112D00               lxi d,stran
502   03E8 2A0100               lhld wboot
503   03EB 19                   dad d           ;hl = sectran entry point
504   03EC D1                   pop d
```


COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
03ED E9             pchl
                ;
                ;
                setdma:                         ; set dma
                        ; 1 -> BC = dma address
03EE 112100             lxi d,stdma
03F1 C3F403             jmp gobios
                ;
                ;
                ;***************************************************************
                ;***************************************************************
                ;***************************************************************
                ;*                                                             *
                ;*       compute offset from warm boot and jump to bios        *
                ;*                                                             *
                ;***************************************************************
                ;
                ;
                gobios: ;jump to bios entry point
                        ;de ->  offset from warm boot entry point
03F4 2A0100             lhld    wboot
03F7 19                 dad     d
03F8 110000             lxi     d,0
03FB E9                 pchl
                ;
03FC 0000       ret$stack:      dw      0
03FE                            ds      32
                loc$stack:
041E            end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

ADIRBCB	03B5	145	168	413#						
ADPB	03B7	166	174	414#						
BASE	0000	436#	437							
BUFPTR	03AD	170	199	207	239	409#				
CHECKPASS	0286	214	253#							
CHECKV	03A7	180	406#							
CPO	0297	262#	266							
CURDMA	03B3	79	260	268	282	284	293	412#		
CURDSK	0392	64	67	81	150	152	389#			
DCNT	03B1	197	225	227	411#					
DFCB	03B9	415#								
DIRMAX	03BB	178	225	416#						
DSKSEL	0152	81	159#							
EMPTY	0398	196	217	220	236	395#				
FTEST	001B	26	37#							
FUNC100	0027	41	58#							
GETEXTA	0349	70	134	343#						
GETHL	0359	162	165	168	175	178	180	181	183	359#
GOBACK	012F	136	140	145#	232	251	275			
GOBIOS	03F4	454	464	474	483	491	512	524#		
GOTONEXT	0023	43#	64	66	76	78	149	151		
HLR	0368	373#	376							
HLROTR	0367	67	190	370#						
I	039F	88	88	209	222	243	399#			
INFO	03A5	63	70	95	405#					
INTO	033D	335#	339							
INTDIV	0338	316	322	332#						
J	03A0	95	107	400#						
JMP1	0006	26#								
LOCSTACK	041E	61	534#							
MAIN0	008E	87#	89							
MAIN1	00AB	87	95#							
MAIN2	00CD	106#								
MAINR	0111	134#								
MAINX0	00DF	118#	119							
MAINX1	00E7	103	118	121#						
MOVE	034F	98	349#	384						
MOVE0	0350	353#	357							
NBANK	000F	31#								
NEW	039D	125	248	397#						
NEWPASS	0395	73	98	392#						
NEXTA	000A	28#	44							
NEXTJ	0009	27#								
NF0	0257	240#	241							
NF1	0262	240	243#							
NF2	026D	245#	246							
NF3	0276	245	247#							
NFCBS	039E	86	187	205	398#					
NOSFCB	0122	91	139#							
NOSPACE	027D	236	251#							
NOTFOUND	023F	228	234#							
OFFSET	03A9	181	317	407#						
OLDPASS	0394	270	283	391#						
OOPS	0236	203	231#							
PASSPROT	0396	393#								

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

PATCH 001A 34#
PREV 000C 29#
RDSEC 03D8 304 479#
READ 0024 442# 482
READDIR 02F3 202 238 300#
READONLY 012A 68 142#
REMOVE 000E 30#
RETSTACK 03FC 60 154 532#
RSXNAME 0010 32#
SAVSECT 03A2 220 237 402#
SCBPB 03BD 78 418#
SCR0 02CD 283 286# 287
SCR1 02E7 294# 295
SCRAMBLE 02BC 100 277#
SDSK 0018 438# 453
SEARCH 01B6 84 194#
SECT 0397 196 219 229 229 237 308 321 325 394#
SECTOR 0319 303 310 320#
SECTRN 03E4 327 496#
SELDSK 03C1 152 161 449#
SERIAL 0000 25#
SETDMA 03EE 200 508#
SETSEC 03D0 328 469#
SETTIME 0393 71 91 390#
SETTRK 03C8 317 459#
SFCB 03A1 93 103 379 401#
SFCBADDR 03A3 95 116 404#
SOFF 03BD 419#
SPACE 0018 33#
SPT 03AF 176 189 191 316 322 410#
SRC0 01CB 202# 229
SRC1 01E0 209# 223
SRC2 01FA 210 217#
SRC3 0212 217 219 221#
SSEC 001E 440# 473
SSET 03BE 420#
ST0 00FF 125 128#
ST1 010D 126 131#
STAMP 0399 75 384 396#
STAMPER 0374 126 131 378#
STDMA 0021 441# 511
STMP 0389 380 381 384#
STRAN 002D 444# 501
STRK 001B 439# 463
SUBDH 035F 227 363#
SVALUE 03BF 421#
TRACK 030A 302 309 314#
WBOOT 0001 437# 502 526
WRITE 0027 443# 490
WRITEDIR 02FD 135 307#
WRONGPASS 02B2 264 273#
WRSEC 03DE 311 487#
XLT 03AB 163 326 408#

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____